EXPLORATORY LEARNING FOR WIRELESS NETWORKING

Thomas Sturgeon

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews

2010

Exploratory Learning for Wireless Networking

Thomas Sturgeon

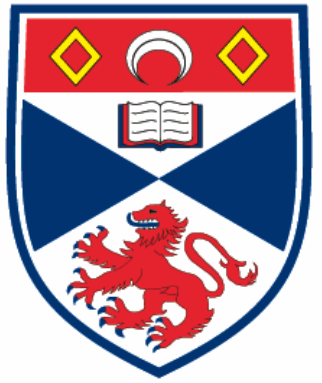
A dissertation submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

of the

University of St Andrews

May 2010



School of Computer Science

University of St Andrews

St Andrews, Fife

KY16 9SX

Abstract

This dissertation highlights the importance of computer networking education and the challenges in engaging and educating students. An exploratory learning approach is discussed with reference to other learning models and taxonomies. It is felt that an exploratory learning approach to wireless networks improves student engagement and perceived educational value.

In order to support exploratory learning and improve the effectiveness of computer networking education the WiFi Virtual Laboratory (WiFiVL) has been developed. This framework enables students to access a powerful network simulator without the barrier of learning a specialised systems programming language. The WiFiVL has been designed to provide "anytime anywhere" access to a self-paced or guided exploratory learning environment.

The initial framework was designed to enable users to access a network simulator using an HTML form embedded in a web page. Users could construct a scenario wherein multiple wireless nodes were situated. Traffic links between the nodes were also specified using the form interface. The scenario is then translated into a portable format, a URL, and simulated using the WiFiVL framework detailed in this dissertation. The resulting simulation is played back to the user on a web page, via a Flash animation.

This initial approach was extended to exploit the greater potential for interaction afforded by a Rich Internet Application (RIA), referred to as WiFiVL II.

The dissertation also details the expansion of WiFiVL into the realm of 3-dimensional, immersive, virtual worlds. It is shown how these virtual worlds can be exploited to create an engaging and educational virtual laboratory for wireless networks. Throughout each development the supporting framework has been re-used and has proved capable of supporting multiple interfaces and views.

Each of the implementations described in this dissertation has been evaluated with learners in undergraduate and postgraduate degrees at the University of St Andrews. The results validate the efficacy of a virtual laboratory approach for supporting exploratory learning for wireless networks.

Acknowledgements

This PhD would never have been completed without the help from the following people. Firstly I would like to thank my supervisors Dr Colin Allison and Dr Alan Miller, without whom this PhD would not be possible.

Thank you to my mum for keeping me motivated and always supporting me during this long, long process.

Thanks to everyone who has worked with me in the office, especially Iain and Kris who have helped me with various problems over the years.

I am indebted to my beautiful Kate for listening to all my rants, keeping me going through the PhD and for reading through this dissertation.

Declarations

I, Thomas Sturgeon, hereby certify that this thesis, which is approximately 53,500 words in length, has been written by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a higher degree.

I was admitted as a research student in October 2005 and as a candidate for the degree of Doctor of Philosophy in October 2006; the higher study for which this is a record was carried out in the University of St Andrews between 2005 and 2010.

Date ……..….. Signature of candidate ……..

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Doctor of Philosophy in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

Date ……..….. Signature of supervisor …….

In submitting this thesis to the University of St Andrews we understand that we are giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that my thesis will be electronically accessible for personal or research use unless exempt by award of an embargo as requested below, and that the library has the right to migrate my thesis into new electronic forms as required to ensure continued access to the thesis. I have obtained any third-party copyright permissions that may be required in order to allow such access and migration, or have requested the appropriate embargo below.

The following is an agreed request by candidate and supervisor regarding the electronic publication of this thesis:

Access to Printed copy and electronic publication of thesis through the University of St Andrews.

Date ……..….. Signature of candidate …….

Date ……..….. Signature of supervisor …….

To my mother and father

Contents

List of Figures

List of Tables

1 Introduction

1.1 *Introduction*

Computer networks are an integral part of modern day life, from mobile phones to Internet access. Governments, academic institutions, banks and many other sectors depend on networks for communications. As such it is hard to overstate the importance of computer networking education.

Networks have a rich dynamism with complex and challenging interactions which should result in learning about them to be an incredibly exciting topic. However, while there is no shortage of good networking textbooks [1-4], the traditional teaching modes of lectures and exercises often fail to engage students taking computer network modules, especially in the crowded curriculum. In network education there is often a rift between learning about protocols from lecture slides and exploring and explaining protocol interactions in a realistic environment. Exploring a protocol gives a better understanding to the student and as such can enable creation and evaluation of knowledge in the area (see section 2.3.6).

There are unique challenges to learning and visualising networks. Network protocols are invisible to the naked eye and for some this can be a difficult and abstract notion. Networks also operate at such a rapid pace that it is impossible to follow an exchange in real time.

Networks can be explored and experimented with in several ways, e.g. simulation, emulation and network-level programming. Simulation is attractive for educational purposes. For example, scenarios can be created that would be infeasible in a computer laboratory, e.g. a several thousand node network with random file transfers between each node. In addition the provision of dedicated hardware facilities can be costly and does not scale with class size. Advanced network simulators, such as ns-2 [5], however, are large, complex pieces of software. They require the comprehension of specialised computer languages and the ability to interpret traces in a specialised format before users are able to focus on learning about a specific networking topic. Another mode of educational network interaction involves programming on a real infrastructure to create packets to observe protocol interactions; a problem with this approach is the steep learning curve required to become a competent network programmer - the student must first attain a high level of proficiency in a system programming language. These barriers may be overcome by dedicated students but will serve to discourage weaker students from the development of a lasting understanding.

The importance of exploratory learning is demonstrated in this dissertation through reference to educational theory. This dissertation details the design and implementation of a set of interactive learning resources, the WiFi Virtual Laboratory (WiFiVL) which is designed to provide anytime anywhere access to a self-paced or guided exploratory learning environment. Whilst the focus is on the IEEE 802.11 protocol, the approach can be applied in other areas of network education.

1.2 *Thesis Statement*

The thesis of this dissertation is that the provision of an exploratory learning environment can engage and enhance a student's learning experience in wireless networking. Evidence in support of this thesis has been shown in:

- Domain of theory – This dissertation discusses educational theory which indicates an exploratory and open-ended approach to learning is important for a detailed grasp of a topic.
- Feasibility – This dissertation details the feasibility of designing and implementing a system capable of meeting the requirements for exploratory learning about wireless networking.
- Demonstration of validity – This dissertation has outlined that the system designed and implemented was successfully deployed and tested. The validity was further shown through the feedback and evaluation with various cohorts of students.

1.3 *Technology Enhanced Learning*

Technology Enhanced Learning (TEL) provides educational resources through the application of technology. A Technology Enhanced Learning Environment (TELE) uses virtual and physical technology to provide a space where learning can take place [6]. The use of technology for enhancing education is increasing, as is evidenced by organisations such as the Technology Enhanced Learning in Science (TELS) [7], journals such as Transactions on Computing Education [8] and special e-learning programmes such as those funded by the Joint Information Systems Committee [9].

The use of computers was initially limited to advanced government research laboratories. Moore's law predicts that the density of transistors that can be fitted on a chip will double approximately every two years [10]. With this ever increasing capability of CPUs and low cost manufacturing, availability of powerful information and communication technology has improved. Computers are becoming more and more ubiquitous with increasingly cheaper and smaller models available to many more people. Everyday life is now dominated by computers in the developed world.

Human-computer interfaces can be categorised as 1, 2 or 3-dimensional. An example of a 1-dimensional interface is the Command Line Interface (CLI). The CLI is a simple though abstract way of interacting with a computer. The CLI allows users to enter commands which are then executed by a command line interpreter. An early example of which is the teletypewriter machines which were connected to mainframe computers enabling a range of commands to be executed [11]. Seminal operating systems such as Unix [12] were originally controlled by the user solely through a CLI. While modern operating systems have removed the need for a CLI, it is recognised how powerful they are for advanced users [13] and as such have evolved alongside the Graphical User Interface (GUI). Current operating systems such as Windows Vista [14], Ubuntu [15] and Mac OS X [16] all include the capability to use a CLI.

Virtual environments were also possible through a 1-dimensional interface. The first virtual environment, Adventure, was entirely text based. Users were presented with a description of their environment and from that they could choose options such as “Left”, “Right” or “Run”. The use of such a virtual environment was expanded to support multiple users where online chat and role-playing missions were provided in the form of text. An early example of which was named Multi User Dungeon (MUD or MUD1) [17] and inspired a genre of games named after it. Examples of MUDs include MIST [18], DikuMUD [19] and SHADES [20].

As operating system capabilities have evolved so has the user interface. The invention of the mouse and addressable frame buffers enabled a 2-dimensional way for users to interact with the computer. An often cited early example is the Xerox Star Workstation which provided a GUI desktop and was the precursor to icons and pop-up boxes. It was used at Xerox and in many universities [21]. The advancement of such interfaces further enabled environments for multiple users and collaborative learning. Examples of 2-dimensional interfaces being used for collaboration can be found in many popular Web 2.0 applications such as wikis, blogs, social networking and bookmarking software. These all enable easy sharing of information from multiple sources and assign meaning and context to user-generated information.

With expanding hardware and software capabilities new immersive environments have been made possible, an example of which is the 3-dimensional Multi User Virtual Environment (MUVE). The concept of a MUVE, as shown by text-based MUDs, has existed long before hardware and software developments could provide us with a 3-dimensional representation.

This dissertation describes the use of 2 and 3-dimensional virtual environments to host a virtual laboratory. The virtual laboratory allows the exploration of the IEEE 802.11 protocol.

1.4 *Overview of the WiFiVL*

The WiFi Virtual Laboratory (WiFiVL) is an exploratory environment where students can create a scenario in which nodes communicate with each other using IEEE 802.11 as implemented in the Network Simulator 2 [5] (ns-2).

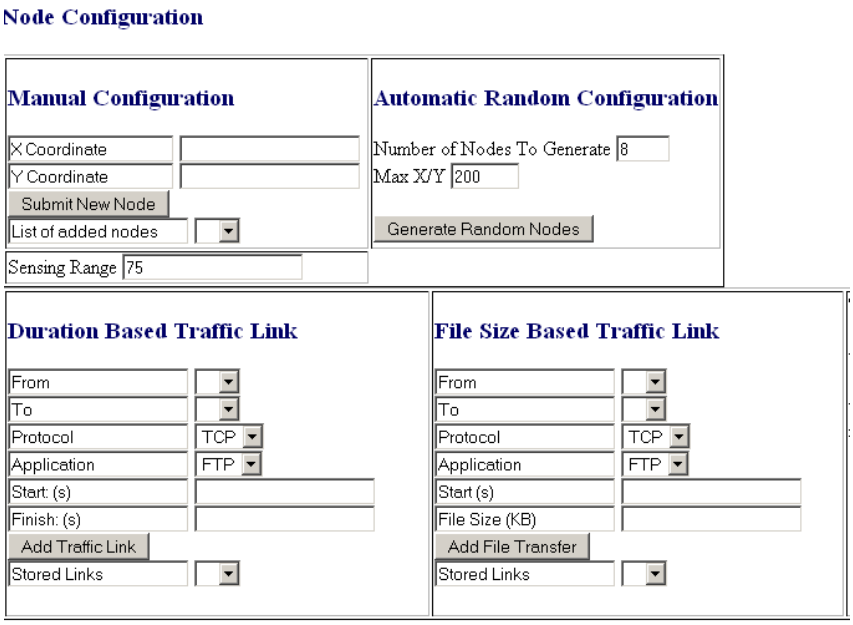


Figure 1: Screenshot of a section of the form based input for scenario description

Initial experiments with creating a wireless virtual laboratory involved the users describing a scenario using a web form as shown in Figure 1. There are form elements for node positioning and traffic link construction. When a user submits the form a URL describing the scenario is generated and submitted to a Java Servlet. The system detailed in this dissertation translates the URL into an OTcl script [22], which is then executed by ns-2. Once executed, the system translates the results into a format (XML) that can be read by a Flash binary (XML) and is then animated to the user. Figure 2 shows the interface created for scenario playback with controls for time dilation, pausing, and a textual trace of events in the network. The animation uses expanding concentric circles and colour codes to represent different signal types and patterns. The duration that the ring remains at its maximum diameter is proportional to the volume of data being transmitted.

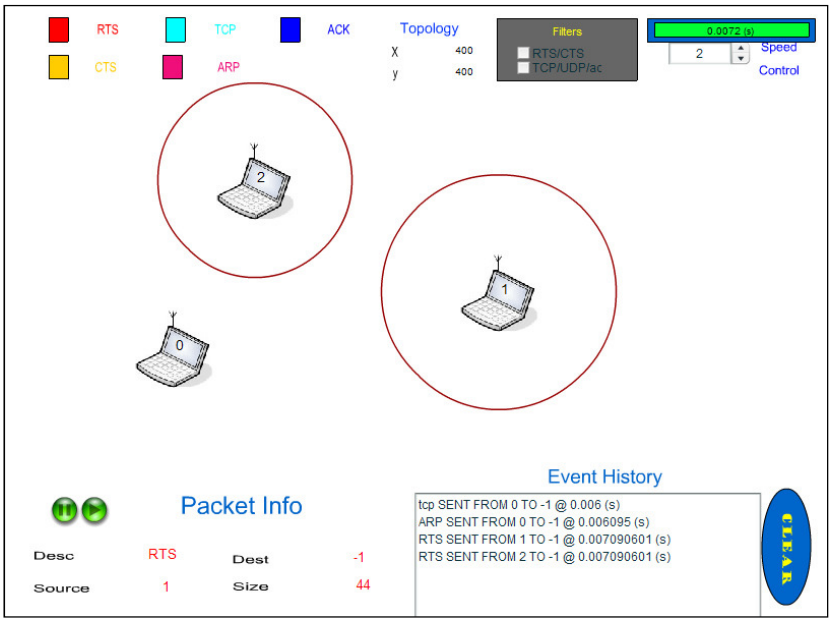


Figure 2: Screenshot of the WiFiVL Flash binary animating two expanding RTS signals

Whilst some users provided high usability scores when using the web-form for inputting a scenario, for others it was too abstract. A container, WiFiVL II, was designed allowing students to construct a scenario using a point and click interface. The same container is used to view the results. WiFiVL II was designed using a Rich Internet Application (RIA) framework (see section 4.8); an example screenshot is shown in Figure 3. When the user constructs a scenario a URL is generated (in the same way as in the previous version of WiFiVL) that is unique to that scenario. This URL is used to interface with the same supporting system as WiFiVL I, allowing extensive code re-use. This means that one can construct a scenario in WiFiVL I or WiFiVL II and review the results in either. This flexible feature allows students to use identical laboratory worksheets then construct and review scenarios in the version that is most suited to them. WiFiVL I and WiFiVL II received positive feedback from undergraduate and postgraduate students in the School of Computer Science at the University of St Andrews.

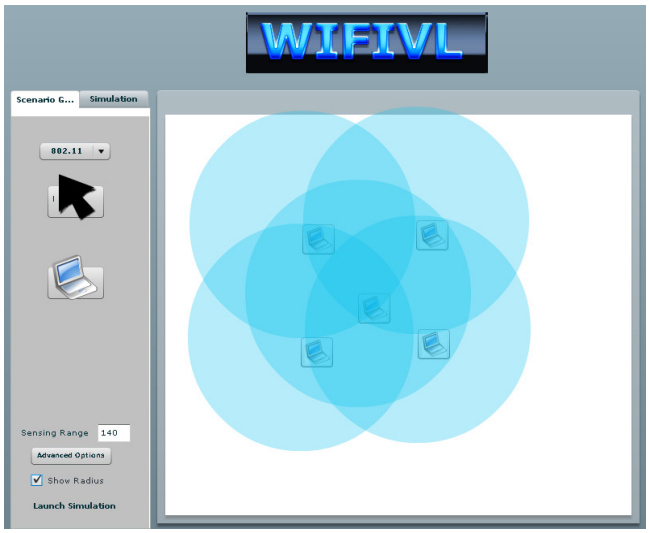


Figure 3: Screenshot of a scenario constructor and animator for WiFiVL II

To research novel interfaces the realm of a Multi User Virtual Environment (MUVE) was explored. The use of MUVEs in education, whilst expanding, is still limited. Educational use is often focused on attempts to replicate conventional teaching methods such as lecturing to a class, providing a slide show on a topic or holding a tutorial discussion (albeit round an interestingly drafted camp fire). The use of virtual worlds for education has been advanced through the development of WiFiVL. A version of the WiFiVL was created in Second Life named WiFi Second Life (WiFiSL) and in Open Simulator named WiFi Open Simulator (WiFiOS). A screenshot of the island developed in Open Simulator is shown in Figure 4.

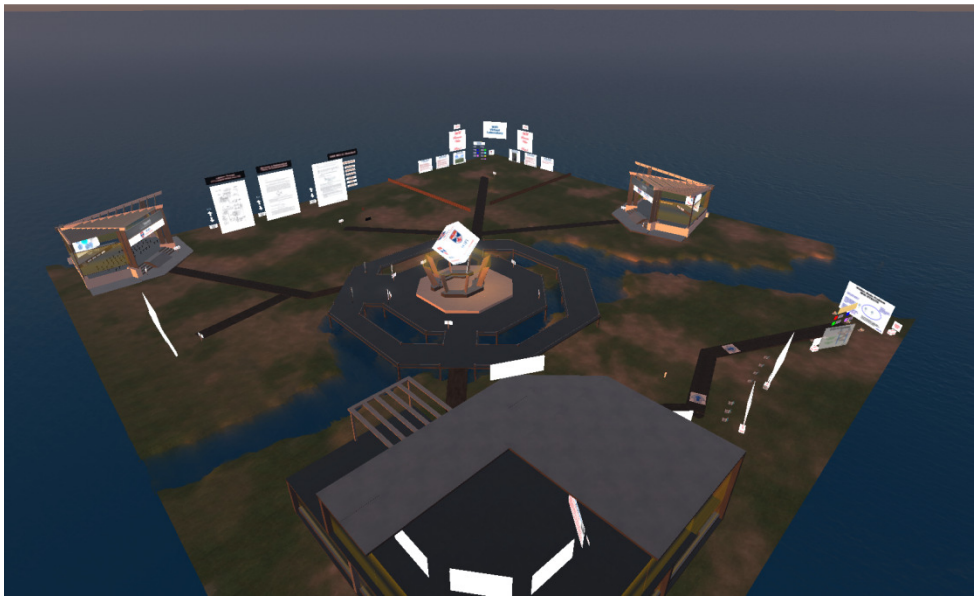


Figure 4: Screenshot of the island developed in Open Simulator (WiFiOS)

An exact translation of WiFiVL I, e.g. a webpage from a browser into a virtual world where an *avatar* fills out a form does not take advantage of the immersive surroundings afforded by a virtual world.

WiFiSL enables students to place networking components from their *inventory* into the virtual world. Once placed in the virtual world, traffic links are constructed between them. When a user has finished constructing traffic links the scenario is described using a URL. This URL is the same type of URL used by both WiFiVL I and WiFiVL II. The results from the users' scenario are displayed back to them using the visual capabilities of the virtual world. An interesting feature of the WiFiSL that was outside the reach of WiFiVL I and II is the use of collaboration. Students can collaboratively create and review a scenario in a MUVE from a variety of perspectives.

1.4.1 System Structure

The system implemented for WiFiVL generates a URL schema which describes a scenario. An overview of the systems interactions are demonstrated in Figure 5.

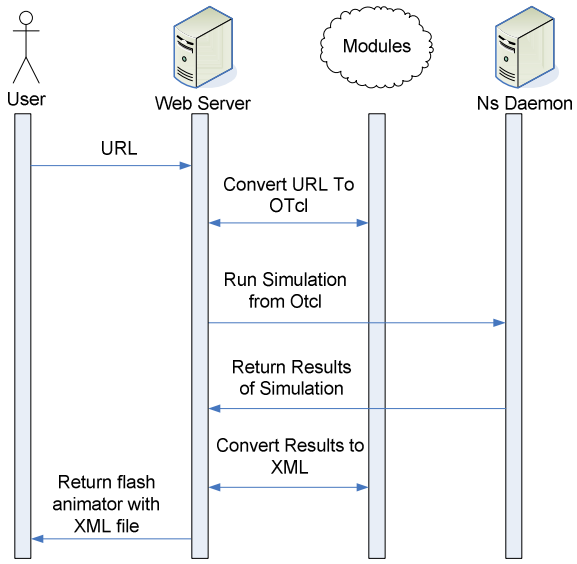


Figure 5: UML sequence diagram of WiFiVL

Each input interface constructs a URL specifying a node location and any traffic links between the nodes. The URL is translated by a Servlet that constructs a Java object which describes the scenario. The Java OTcl Build API (jOBA) was developed for WiFiVL and enables a scenario to be converted from a Java object instantiation into an OTcl script. An ns-2 daemon was developed to manage the execution of the OTcl script in ns-2 and to return the results.

Once the results have been returned from the ns-2 daemon another component translates this into XML which can be read by the Flash binary. There are alternative outputs which enable the results to be streamed to a bandwidth limited environment such as Second Life. The system is discussed in detail in Chapter 5.

1.5 *Contribution of Dissertation*

- This dissertation reports on the design, implementation and evaluation of a framework that supports exploratory learning for wireless networks. The framework provides realism, collaboration and a space in which students can explore, hypothesise, experiment and evaluate wireless protocols through scenario construction.
- A review of related work of virtual laboratories and network simulators that may support exploratory learning has been carried out, summarised. A comparison is also provided.
- The key educational points of the IEEE 802.11 standard, such as those commonly addressed in text books are identified. Their relevance is demonstrated within the WiFiVL framework.
- When a simulator is used it is often either too simplistic to be realistic or too realistic to be useful. The WiFiVL framework strikes a balance between realism and complexity to provide a usable, engaging and educational environment. This dissertation details an implementation that enables ns-2 or future alternative network simulators to be used in a virtual world or in a simple webpage without requiring detailed knowledge of specialised programming languages.
- The WiFiVL framework has been written and tested in Java with detailed documentation provided. Supporting materials include a description of the technologies used to create, maintain and test the WiFiVL. Extending from this framework are four comprehensive applications: WiFiVL I, WiFiVL II, WiFi Second Life (WiFiSL) and WiFi Open Simulator (WiFiOS). The source code that implements the architecture is structured, component-oriented and designed for reusability.
- Students at various levels of University education have used and evaluated the software. Formal evaluation techniques have been used such as interviews and questionnaires about perceived educational value and usability. The resulting evaluations have been reviewed and discussed. The use of these evaluation techniques has indicated that the development of the WiFi Virtual Laboratory, from web forms to virtual worlds, has provided increased usability, greater engagement and greater perceived educational value to students.
- The user evaluation of the virtual worlds showed a positive response and the publications demonstrate an interest from the wider academic community.

1.6 *Selected Publications*

A WiFi Virtual Laboratory. Sturgeon T., Miller A., Allison C. In *Proc. 7th Annual Conference of the Higher Education Academy Subject Centre for Information and Computer Sciences, Dublin*, pp 207-211, 2006.

Exploratory Learning for Computer Networking. Allison, C., Miller, A., Getchell, K., and Sturgeon, T., in ICWL 2007, *Lecture Notes in Computer Science,* vol. 4823/2008, pp. 331-342, 2008.

802.11 Wireless Experiments in a Virtual World. Sturgeon, T., Allison, C., and Miller, A. *ACM SIGCSE Bull.* 41, 3, pp 85-89, 2009.

Exploring 802.11: Real Learning in a Virtual World. Sturgeon T., Allison C., Miller A. In: *Proc. 39th IEEE International Conference on Frontiers in Education, San Antonio, Texas,* pp 907-912, 2009.

1.7 *Dissertation Roadmap*

Chapter 2 provides an overview of learning models including exploratory learning. Chapter 2 also provides a review of learning taxonomies and their applicability to Computer Science. The use of exploratory learning, its importance and how it relates to WiFiVL is provided in the summary.

A review of related work is provided in Chapter 3 along with terms of reference. The terms of reference are used to categorise the various educational tools available. The first section in related work focuses on educational tools, their usage, structure, similarities and differences relating to this work. A taxonomy is provided for tools that are in use in the computer networking education domain. Further sections explore other areas of study that have used exploratory learning tools. The related work chapter also details educational projects that have made use of a Multi User Virtual Environment (MUVE).

Chapter 4 contains a description of the technologies used to create the WiFiVL. Different types of simulation techniques and implementations are discussed. A summary of the usability heuristics is provided. These are referenced throughout this dissertation when discussing the design of the WiFiVL. The implementation of WiFiVL utilised Servlet technology and a mixture of other programming languages. The work detailed in this dissertation focused on an educational exposition of the IEEE 802.11 protocol and as such a clear description of the key features of the protocol is provided. Whilst IEEE 802.11 is the focus in this instantiation of the framework, the system is designed in such a way that other wireless protocols could be taught about and experimented with using this virtual laboratory approach.

Chapter 5 provides details on the infrastructure created to support WiFiVL I. Details of the system architecture and implementation are provided. An alternative user interface was created using a RIA and its design is also detailed in this chapter. User evaluation techniques, results and analysis are provided for both WiFiVL I and WiFiVL II.

Chapter 6 details the progression from 2-dimensional interfaces to 3-dimensional interfaces that make use of virtual worlds. A range of educational scenarios provides information on the types of activities that a virtual world can take advantage of. The architecture for enabling a virtual laboratory *in-world* is given through the decomposition of its components. Because of the novel interface a discussion is provided on both visualisation considerations in 3-dimensions as well as user interface design for scenario construction. Several evaluation sessions took place with a virtual laboratory in Second Life. The results are discussed along with the suitability of Second Life. The instantiation of the WiFiVL in Open Simulator (WiFiOS) is then evaluated using Computer Science undergraduate students at the University of St Andrews. The evaluation results are provided and discussed.

1.8 *Summary*

This chapter has highlighted the importance of learning about wireless networking. The limitations of conventional teaching methods and materials have been listed, and the barriers to supporting an exploratory approach have been identified. These barriers include the learning curve required to access a modern network simulator, the disengaging use of textbooks and the difficulty in learning a programming language to a high level of proficiency required for network programming. An overview of the dissertation has been provided giving a description of the main work conducted during the study period. The main work has involved the creation of a framework which supports exploratory learning for wireless networks in several environments. Selected publications and a summary of contributions have been listed.

2 Approaches to Learning

2.1 *Introduction*

This chapter provides a review of some taxonomies and models that have been constructed to classify and characterise the learning process. The classification of learning provides an approach to structuring a course where the student is expected to attain a high level of proficiency. A taxonomy can also be used to show a student's achievement in the topic area and the path to the final goal. This chapter includes the classification of learning that has been used across several domains of learning.

There are a range of models for how a student can achieve the learning goals outlined in a taxonomy. Section 2.3.6 provides a formal definition for a variety of learning models. These models range from didactic, where the learner is instructed by a teacher, to the situation where the learner acquires the knowledge through guided or unguided exploration.

2.2 *Learning Taxonomies*

A taxonomy is an ordered classification system that provides a way of comparing different items and ordering in a meaningful way. It provides a meta-description for "understanding about understanding" [23]. An example of a successful taxonomy is the Linnaean Taxonomy [24] created in 1760; it has undergone large changes but is still the predominant way of classifying living organisms in biology.

2.2.1 Bloom's Taxonomy and Andersons Revisions

Learning taxonomies have been created to help define not only what the student should learn from an exercise but also what cognitive level should be achieved. At the lower end, learners can be at a stage where they can recite but not understand something. At the higher end the student can defend, expand upon and apply the knowledge in novel situations. This is evident in Bloom's taxonomy, which splits learning into affective, cognitive and psychomotor domains [25].

- **Affective** – Emotional skills and growth, ability to empathise etc. (attitude). There are several differing levels of achievement.
    - Receiving – The student is willing to pay attention
    - Responding – The student is stimulated by the material and provides reaction and input on topics
    - Valuing – The student attaches value to the topic of learning
    - Organizing – The student can compose varied values and ideas, incorporating these into their understanding and building upon what has been instructed and learnt
    - Characterizing – The student's internal model affects their behaviour

- **Cognitive** – This is the most widely used and quoted aspect of Bloom's taxonomy and refers to knowledge acquisition, comprehension and expansion to provide the learner with a full understanding of the problem domain.
    - Knowledge – The act of recall, the student has memorized by rote a particular piece of information and is able to recall it when required. This is seen as the lowest step in learning. However, Bloom's taxonomy can be seen as ordered steps to learning and as such each step is important.
    - Comprehension – Demonstration of understanding the material. This is a clear step up from the simpler recall level of knowledge. A requirement here is the ability to show an understanding. Such activities would include paraphrasing and giving examples.
    - Application – The ability to bring to bear knowledge of previous domains and apply it in a new situation. This step can involve the novel application of an understood theorem in a different unanticipated domain.
    - Analysis – The capacity to deconstruct the information and provide insight into the principles underpinning the information. Analysis also involves recognising relationships between information and the inference of related material or ideas.
    - Synthesis – Development of wholly new ideas derived from the understanding of all the previous levels. An example of synthesis is developing a new theorem, which would require detailed understanding of the domain, other knowledge available and a grasp of the underlying principles.
    - Evaluation – The ability to provide a judgement on a set of information and be able to defend and explain that judgement.
- **Psychomotor** – Bloom intended this domain to focus on the physical manipulation of objects but did not complete the work. There are various proposed structures for this domain [26-27]. Below is an example structure by Dave [28].
    - Imitation – Observation and repetition
    - Manipulation – Certain autonomous actions
    - Precision – Cultivation of previous stages and reduction of mistakes
    - Articulation – Coordinating several actions
    - Naturalisation – Fluid movements without error and which appear to flow naturally from the student

Bloom describes learning using the above three domains and implies that the use of higher levels requires the understanding and command of the lower levels. As such, it can be seen that Bloom's taxonomy can be used for structuring the design of a curriculum.

Critics of Blooms cognitive taxonomy view learning as not requiring a strict hierarchical format. Anderson, a student of Bloom, proposed revisions where the lower levels of learning are retained but the higher three (analysis, synthesis and evaluation) can be attained in no fixed order and as such are parallel rather than hierarchical [29]. An additional revision to Bloom's taxonomy is the reordering of evaluation and synthesis and the use of verbs rather than nouns for each level. The revised taxonomy proposed is summarised in Table 1.

Analyzing	Evaluating	Creating
Applying		
Understanding		
Remembering		

Table 1: Revision to Bloom's taxonomy without a strict hierarchical format

Anderson viewed Bloom's taxonomy as having a one-dimensional view of learning [29]. Anderson proposes a two-dimensional view of learning with the first being the knowledge dimension, defined as the type of material to be learned. The second dimension is the way in which that material is learned, that is the revised list provided for cognitive learning (remembering, understanding etc.). The knowledge dimensions introduced are [30]:

- **Factual** – Building blocks of learning, the basic elements of a domain of learning.
- **Conceptual** – Interrelationships of the building blocks.
- **Procedural** – How to do something, the techniques required in the given domain and the appropriate application.
- **Meta-Cognitive** – Self-awareness of knowledge, e.g. the learner recognising their own strengths and weaknesses in a domain.

This is represented in Table 2 [30].

Knowledge Dimension	**Cognitive Process Dimension**					
	Remember	Understand	Apply	Analyse	Evaluate	Create
Factual	List	Summarise	Classify	Order	Rank	Combine
Conceptual	Describe	Interpret	Experiment	Explain	Assess	Plan
Procedural	Tabulate	Predict	Calculate	Differentiate	Conclude	Compose
Meta-Cognitive	Appropriate Use	Execute	Construct	Achieve	Action	Actualise

Table 2: Bloom's revised two-dimensional taxonomy and example verbs

An example of this revised taxonomy can be seen in the educational journal Theory into Practice [31]. In [32] an English and History course entitled "Western Culture" was revised and constructed using Anderson's revised version of Bloom's taxonomy. The conclusions were that the use of the revised Bloom's taxonomy helped communication between differing fields. The use of Bloom's revised taxonomy also enabled a revised outlook for assignments and assessments, allowing students to operate and display more complex levels of thinking.

2.2.2 SOLO

The Structure of the Observed Learning Outcome [23] (SOLO) taxonomy provides a set of categories which analyses the student's answers to the topic material rather than the material itself. The different SOLO levels are:

- Pre-structural – Misunderstanding of the area; answer is unrelated to the area
- Uni-structural – Simple response showing superficial understanding of an area
- Multi-structural – Lacking consistency and an underlying understanding though provides a selection of responses technically correct
- Relational – Understands the concepts involved and is capable of applying this knowledge in a familiar setting
- Extended abstract – In-depth understanding of the area and is capable of applying this in unfamiliar and new settings

Knowledge and use of the SOLO taxonomy is expanding as more evaluation results are published [33-35].

2.2.3 ACM ITiCSE Taxonomy

Computer Science has differing requirements from the arts community where critique may be valued higher in certain areas, such as classical reading.

A working group at ITiCSE (Innovation and Technology in Computer Science Education) published a paper on the weaknesses of the previously discussed taxonomies in this chapter and an approach to a better design [33]. This working group focused on the construction of a learning taxonomy specifically for Computer Science as it has some unique requirements. The working group highlights a common problem where students can understand code but cannot write or debug a program. A proposed revision of Bloom's taxonomy was described to differentiate between reading and writing code giving a 2-dimensional matrix as shown in Table 3.

Create				
Apply				
None				
	Remember	Understand	Analyse	Evaluate

Table 3: A representation of the ACM ITiCSE working group for a revision of Bloom's taxonomy for Computer Science

The ideal for a student is to progress from the bottom left hand corner of "remembering and none" to the "evaluate and create" top corner. As this is a 2-dimensional approach, it recognises that students learn in different ways and as such there are different routes to the ideal of create and evaluate or "Higher Application". This approach satisfies a criticism of Bloom's taxonomy that it is too rigid in its hierarchical approach to learning.

Table 4 and Table 5 are taken from [33] and show examples of the 2-dimensional matrix highlighting various learning paths of students.

Create				
Apply				
None				
	Remember	Understand	Analyse	Evaluate

Table 4: A reconstruction from the ITiCSE working group showing the 2-dimensional matrix taxonomy for a student with theoretical competency only [33]

Create				
Apply				
None				
	Remember	Understand	Analyse	Evaluate

Table 5: A reconstruction from the ITiCSE working group showing the use of the 2-dimensional matrix displaying a student who is unable to progress past a trial and error approach [33]

2.3 *Learning Models*

2.3.1 Didactic Model

The didactic model of learning is a traditional one in which there is a teacher and a student. The teacher has the expert knowledge and imparts this to the student. This model of learning is focused on the lower levels of Bloom's learning taxonomy which involves remembering, understanding and applying.

2.3.2 Collaborative Model

The collaborative model of learning is when a group of learners combine their efforts and attention with the aim of completing a common goal. There are many activities where collaborative learning can be used, e.g. writing and programming. The wide variety of new communication methods available online provide a tool for many geographically remote groups to collaborate in a common goal. A key example of online collaboration is Wikipedia [34]. There have been 13 million articles written on Wikipedia by using a simple web browser text editor and mark-up language by a group of volunteers. The Wikimedia Foundation [35] has organised its contributors through self-policing to avoid defacements and other attacks. These strategies construct an environment of group efforts but with individual accountability. Other Web 2.0 tools such as blogs provide a central location for a writer to publish work which can then be easily commented on by readers and syndicated.

Social bookmarking services Delicious [36] and Reddit [37] allow users to bookmark sites of interest and provide tags that describe the information. Tags and site-friendships can be used to explore similar areas of interest, providing a collaborative environment. Other prevalent tools available are the use of joint whiteboards [38], instant messaging and video-conferencing [39-40].

A generic name for the use of online resources for collaboration is Computer-Supported Collaborative Learning (CSCL). Communities can be established around such collaborative efforts and provide an environment of achieving a common goal. The self-evaluation that each participant performs before submitting their own work and when reviewing that of others facilitates the development of critical thinking [41]. The use of the collaborative learning model reflects the constructivist approach discussed later in this section.

There are many techniques using the collaborative model out with CSCL such as the Think-Pair-Share approach [42]. A leader assigns a question to a group of learners, the learners divide into pairs and reflect on the question and provide joint answers. Learners using this approach reported lower anxiety when attempting to solve a task as they could discuss it with a peer privately first. Even through such a simple example it is clear that there are many benefits to be had in reducing anxiety and enabling critical collaborative thought.

2.3.3 Problem-based Learning

An example of problem-based learning is where a group of learners are given a task, and through collaboration with and guidance from a helper, they arrive at a solution. There is a range of support available to the helper and this may range from zero to full coaching. Some studies show that where minimum guidance is provided the learning is ineffective and inefficient [43]. Initial guidance may be high, then decrease as the students gather a more thorough understanding of the problem domain [44].

The cognitive load a learner encounters may interfere in the primary goal of learning during problem solving [45]. The cognitive process of recognising the current state, the goal state, the steps to get there and any other sub-goals may overwhelm the learner. In some cases the emphasis on problem solving

may inhibit the development of expertise in the topic. However, problem solving has been shown to be an effective means of learning but may require some modification.

2.3.4 Constructivist Model

The constructivist theory proposed by Piaget [46], Glasersfeld [47] and Kant [48] is one in which students can gain knowledge through experience. This knowledge does not have to be gained through the didactical process between learner and teacher commonly associated with education. Constructivism is about how learning occurs and proposes active learning as an important and over-looked approach.

In [49], studies are shown that children grasp substantial knowledge of their surrounding physical world before it would be possible to learn such things from their social and cultural community. Vygotsky gives an example of a child wanting to ride a horse but who is unable to do so. A child over a certain age is capable of imagination and can use a stick to represent the horse, enabling the child to ride the horse in their mind. Through play their imagination is realised and is how they understand the world. Interestingly it may be said that "children's play is imagination in action can be reversed: we can say that imagination in adolescents and schoolchildren is play without action" [50].

Vygotsky constructed the idea of the "Zone of Proximal Development" (ZPD). ZPD is defined as "the distance between the actual development level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers" [50].

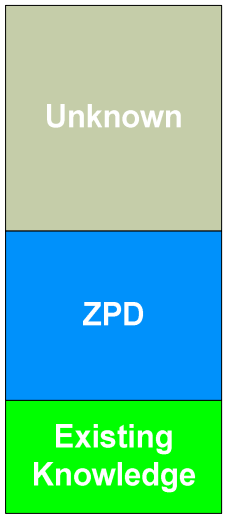


Figure 6: Representation of the Zone of Proximal Development

As more guidance is given by the teacher to the student the ZPD is able to expand and enable the student to reach more of the unknown problem domain. Figure 6 shows that the student has an existing base of knowledge and is capable of attaining all of the ZPD in blue. In Figure 6, the unknown

section is beyond reach. With guidance and knowledge, expansion the whole ZPD section can become existing knowledge and is subsequently understood by the student.

2.3.5 Constructionism Model

Constructionism is an extension of constructivism and theorises that effective learning happens through the creation of tangible objects in the real world. In [51] Papert detailed research where children would program in the Logo language and that the mathematical ideas were as natural as learning French. Papert envisioned a revolution in the way that education was thought of and advocated the advancement of micro computing as a catalyst enabling educational aspiration. A recent example of enabling constructionism is the LEGO Mindstorms Robotics Invention System [52]. Students learn through experiential learning and experimentation. This kit enables students to design, create and attempt to solve various problems with bricks to make a tangible object in the world.

Constructionism and constructivism appear at first glance to be overlapping in their ideas. Both Papert and Piaget are constructivists in their view that children build a model of the world through interactions and that this is a logical and "right" view of the world. As children experiment with their interactions with the world, knowledge is constructed and reconstructed. Both ideas are developmentalist in that knowledge acquisition is incremental and both are interested in how and when a learner will modify their view of the world and build a deeper understanding of it. It can be viewed that Piaget is more focused on the construction of a stable internal view of the world while Papert concentrates on the dynamics of change [53].

2.3.6 Exploratory Learning

This section explores the varying definitions and examples of exploratory learning in networking education and its wider application. In [54] an experiment is constructed to define exploratory learning and any benefits. Their initial definition of exploratory learning states that *"Information is not given to learners in an expository way but an open learning environment is offered in which learners have to formulate themselves: principles, procedures, or higher order skills. By means of inquiry, scientific discovery, problem solving, inductive reasoning and so forth, learners have to acquire their knowledge in an active, constructive way."*

The test subjects were students in mechanical engineering and control theory at the Eindhoven University of Technology. The subjects were put in groups of two, with half of the subjects receiving the original guided assignment and the other half receiving a modified, unguided assignment. Data was collected using thinking aloud[1] protocols, notes made by the tutor, system log analysis of program input and output. These experiments guided Njoo & Jong to defining the exploratory learning process as consisting of four main categories: transformative, regulative, operative and general. The

[1] A protocol where students verbalise their thought process

transformative process is a scientific inquiry and is broken down as: analysis, hypothesis generation, testing and evaluation. Regulative processes refer to the planning, monitoring and general control of the experimentation. Operating the simulation is concerned with the user interface of the simulation program. The general category describes activities such as calculating, making notes and off-task remarks.

Rieman's study [55] suggests that task-oriented exploration was the best method for learning as users reported exploring non-task related areas as inefficient. Rieman uses as an example secretaries requirements for repeated training when new versions of word processing software are introduced into the work place. As new upgrades are made available on a regular basis for software, continual retraining is impractical. It is clear that users today will explore and interact with the software to learn its features without necessarily reading a manual (though this may be used to supplement their knowledge and exploration). This technique of informal, unregulated exploratory learning proves itself to be a good way to build confidence and knowledge of a system. Rieman's definition of exploratory learning is "instead of working through precisely sequenced training materials, the user investigates a system on his or her own initiative, often in pursuit of a real or artificial task".

Rieman investigated how different users approached learning and the use of a particular software. One evaluation technique was to provide a "Eureka Form", which is used to record the completion of a task and how the user completed it (e.g. read the manual, experimented, copied earlier observations of others, asked for help). The distribution of these Eureka slips heavily favoured the exploring method over any other. Another evaluation technique was to provide an activity log where users supplied information about how they spent their day that was updated half hourly. Users of the software were interviewed on a daily basis and the total time spent on non-goal oriented exploring was measured. The measurement of such an activity is very difficult because the users were not told to detail this activity for fear of compromising the results. Similarly the user-conception of non-goal-oriented exploration as being inefficient or wasteful could lead them to not want to report this to the interviewers for fear of being seen as lazy or unfocused. Indeed, Rieman estimates that 60% of users' times in offices are spent on non-goal focused activities. However, how much of this is spent on exploring rather than sending personal emails, for example, is unclear. One interviewee did not like "reading through all that stuff [the manuals] and trying to find the answer to my problems." This user preferred demonstrations of software and would only consult the manual as a last resort.

The research in [56] focused on users' capability to master a system without a manual. In [56] a toy truck with a computer interface was provided without instructions. Users managed to understand most of the functionality, but some of the more advanced features remained unattained, although the reason for this may have been due to the poor interface provided by the truck.

The psychologist Max Wertheimer in his influential book "Productive Thinking" [57] links understanding to transfer, stating that the understanding provides the ability to generalise the solution of one problem and apply it to another. In [58] Lewis describes his new way of providing such a

linkage through his technique of combining causal analysis and analysis-based generalisation named EXPL. Distinctions are made between superstitious methods, where aspects in an example that are not understood are retained with generalisation built from it and a rationalistic method where only the understood aspects are retained.

Designing a suitable interface is an important factor in enabling exploratory learning. In [59] the exploration of software is defined as a user setting a goal and then exploring that domain. The user will select a label most appropriate for their task, and then if positive feedback is received the user will continue to search that domain, in effect a goal search-action cycle is carried out. The poor layout of an interface seriously hinders a user's ability to properly explore the environment and makes the attainment of their goals impossible.

Many sources referenced in this section are not directly concerned with computer network education but the underlying theory and principles are applicable. The previously discussed work emphasises the benefits of exploratory learning.

The definition of exploratory learning for the purpose of this dissertation is allowing students to learn through non-prescriptive methods. An environment is provided in which the student can manipulate the state of something to gain a better understanding of that environment. The user can pursue a specific goal or simply explore the environment. The learning is not wholly without guidance as a general worksheet is provided to help students formulate, test and evaluate a hypothesis.

2.4 *Conclusion*

Approaches to learning can be characterised through taxonomies and models. Early learning taxonomies include SOLO and the work by Bloom and Anderson. There has been an evolution of taxonomies in computer science to show that learning is not a one dimensional process. In the ACM ITiCSE taxonomy the learning process is shown as requiring the student to progress through remembering to actively understanding a concept through its application and the creation and evaluation of a hypothesis derived from it. This requires an open-ended environment suitable for self-paced solo or guided learning.

This chapter has also included a review of learning models. These models include didactic, collaborative, problem-based, constructivist, constructionism and exploratory learning. The models in this chapter range from the students remembering information by rote with no interaction to those where the information must be attained wholly by the student in a supported environment. Concepts found in higher level learning, e.g. creating, analyzing and creating, would seem to be supported by exploratory learning. The exploratory models are seen as enabling a richer understanding of a topic and have strong parallels with the ACM ITiCSE taxonomy. What is clear is that an exploratory learning environment for self-paced guided or unguided learning can be a valuable resource. An exploratory learning environment is well supported by a real or virtual laboratory. A virtual laboratory provides an open space to enable learners to freely explore a topic, test and evaluate a hypothesis. As

such this dissertation details the approach of using a virtual laboratory for learning aspects of the wireless protocol IEEE 802.11.

3 Related Work

Exploratory learning environments allow students to follow their own learning trajectories through interaction with suitable educational tools. Experimentation is a specific type of exploratory learning and an approach we seek to inculcate in all scientific discoveries. Accordingly a laboratory, real or virtual, is a useful facility for both self-paced and structured sessions. Lasting understanding is rarely achieved through a single illuminating experience – it more typically involves crossing the line between puzzlement and comprehension on several occasions and from different perspectives. One of the benefits of exploratory learning environments is that they can be revisited as often as the learner wishes. This chapter reviews work where an environment is made available with the potential for use for exploratory learning in a higher education environment.

3.1 *Taxonomy and terms of reference*

The taxonomy constructed for this dissertation categorises the related work by key features and characteristics. The characteristics defined below are summarised in Table 6.

- Animation – simple pre-determined simulation is conveyed to the student. The type of information being animated may vary from simple interactions to complex real world situations. Interactions shown are generally pre-determined and do not allow extensive and or rich user interaction.
- Simulation – this is when a network simulator is used to generate data resulting from a given scenario.
- Emulation – this describes results that are derived from real world network installations
- Licensing – licensing decrees the amount of access given to the source code for a tool. The term 'Free' is used to refer to projects where a license similar to the GNU General Public License has been issued. GNU General Public License [60] is an example of one of many licences which are intended to allow open and fair usage of software. Some simulators are commercial and require significant costs for licensing. Certain tools require a lecturer from the University to register their details for the group wishing to access the tool. This has been termed as 'Restricted Access' in the taxonomy table. Some tools will allow the use of the website exclusively for education and do so without putting specific GNU licensing on the tool. In such a case the 'Free to Use' term is used.
- Domain – refers to the area being addressed by the tools, e.g. a network simulator, a tool for network structures or the tool is aimed exclusively at the IEEE 802.15.4 protocol.
- Target Audience – this describes the intended users, e.g. research institutions, undergraduate or postgraduate students.

- Delivery Platform – tools may be restricted to an Operating System (OS), a specific language or through a web browser interface.
- Last Update – the frequency with which a tool is updated can be considered as quite important as this can help bring new features or protocols as they are ratified.

	Animation	Simulation	Emulation	Licence	Domain	Target Audience	Delivery Platform	Last Update	URL
JASPER	✓	✓		GNU GPL	User definable to single layer	Undergraduate	Java Applet	15/07/2006	[61]
Emulab		✓	✓	Restricted access	Full networking stack	Postgraduate+	OTcl with form interface	Current	[62]
Opnet	✓	✓		Commercial	Applications and Servers	Commercial			[63]
Cnet	✓	✓		GNU GPL	Full networking stack	Postgraduate+	Linux with TCL/TK	07/03/2006	[64]
Weblan Designer	✓			Free To Use	Wired and wireless architecture	Secondary School	Web Page	20/09/2006	[65]
ReMLab			✓		Campus use only	Undergraduate+	Web Page	2003	[66]
SensaSim	✓	✓		GNU LGPL	Wireless Sensor Networks	Undergraduate+	Java Applet	Current	[67]
iNetwork	✓	✓		Commercial		Postgraduate+	Win 2K/WinXP		
ns-2	✓	✓		GNU GPL	Full networking stack	Postgraduate+	Linux with OTcl	Current	[68]
WiFiVL	✓	✓		Free To Use	IEEE 802.11	Undergraduate+	Web Page	Current	[69]
IREEL			✓	Registered Access	Full networking stack	Postgraduate+	Web Page	Current	[70]
MASSIVE			✓	Unavailable	Wireless		Linux	2005	
Graphical Network Simulator 3	✓		✓	Commercial (Cisco)	Routing	Commercial	Windows, Linux Mac OS X	2008	[71]
Planetlab			✓	Registered Access	Networking	Postgraduate+	Linux Administration	Current	[72]
jFAST	✓			GNU GPL	Finite State Machines	Undergraduate	Java	29/05/2006	[73]
VDE			✓	GNU GPL	Networking	Undergraduate+	Linux	26/07/2010	[74]
VNS	✓		✓	GNU GPL	Networking	Undergraduate	Python	23/04/2010	[75]

Table 6: Related work and key features of systems with the potential for use in exploratory learning

3.2 *JASPER*

Java Simulation of Protocols for Education and Research (JASPER) [76] is a tool for teaching protocols through the animation of a finite state machine. The tool is capable of running in a web browser as a Java applet. JASPER can be used to simulate Alternating Bit, TCP, TFTP, HTTP and BOOTP [77]. JASPER is available with a GNU General Public License [60] which enables editing of the source code. Protocols can be added for animation, though recompilation is required. Figure 7 is a screenshot of JASPER and shows a clear and easy to understand representation of the protocol interactions. The tool includes a textual reference of past events in a text box contained at the bottom.

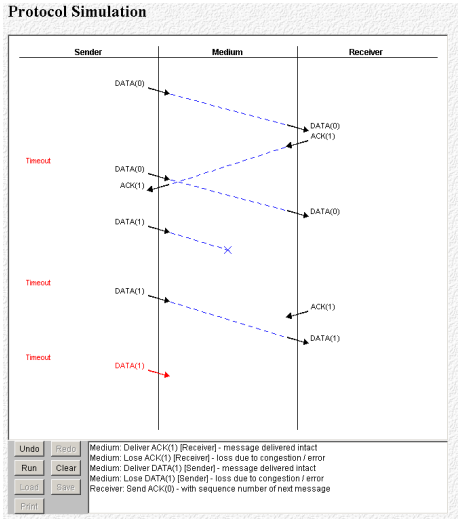


Figure 7: Screenshot of JASPER animating a protocol [61]

A drawback to JASPER is that the user has no control over factors that affect the protocol, e.g. the loss of a specific signal. The simple presentation of the protocol conveys the current activity of the network. More detailed information is lacking intentionally to avoid cluttering the display. JASPER received favourable evaluation through user feedback in the form of questionnaires [78].

3.3 *EmuLab*

EmuLab [79] is a network test-bed that has physical WiFi and wireless sensor network [80] nodes deployed of type IEEE 802.11 a/b/g throughout a building. This deployment enables testing in a real environment. EmuLab is similar to PlanetLab in as much as it is a real deployment of nodes, accessible over the Internet for research. The difference between the two projects is that PlanetLab is distributed across the globe whereas EmuLab is deployed at a single site.

To use EmuLab, a request must be made by a senior member of staff from the requesting institution. There are various etiquettes to accessing nodes in order to avoid interference on a running experiment. A Java GUI is being developed to allow the construction of experiments.

The user is located somewhere online and connects to a central point of the EmuLab network. A portion of the machine is used to make traffic connections. The connections between the machines are configured and then the scenario is run. A scenario is configured for EmuLab using OTcl. Emulab allows gathering of real world results and provides an ad-hoc XML-RPC interface to swap experiments in and out. Emulab is aimed more at research than for undergraduate education and has a steep learning curve of programming languages, configuration and etiquette that must be mastered before using the test-bed.

3.4 *Opnet*

Opnet [63] is a commercially available network simulator that has been used in universities for teaching advanced networking courses. This is a proprietary simulator and formal applications must be made for a university license. There are examples of this simulator being used for teaching Voice Over Internet Protocol and wireless networks to Masters students [81-82]. Opnet is suited to advanced users such as those in postgraduate courses. Due to its closed nature, it is difficult to compare this tool with others in this chapter. The lack of open licensing has hindered its widespread adoption and usage in education.

3.5 *CNet*

CNet [83] is a simulator designed to expose interactions at multiple layers (data-link, network, routing and transport layer) and runs exclusively on UNIX systems. Network protocols must be written in the ANSI-C programming language which, due to the complexity of the language, can create a higher entry barrier for interacting with the network. CNet is provided under the GNU General Public License [60], with its source code, make-files and documentation freely available from the website. A graphical representation of the network can be viewed via TCL/TK or an ASCII representation may be provided as well. CNet can be configured through the command line or its own GUI.

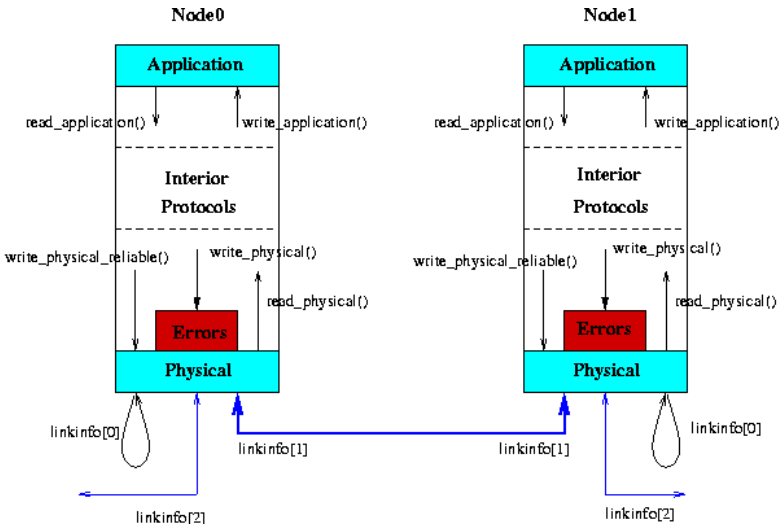


Figure 8: The CNet simulation model [64]

The nodes in CNet can be connected either by point-to-point links or Ethernet segments and can exist as hosts or routers. Figure 8 shows that each host contains the Application, Errors and Physical layers with multiple numbered links between hosts. Hosts are the equivalent of workstations with an OSI model approach allowing Application Layer to communicate with an Application Layer on a remote host. A physical layer is provided that can be modified at runtime through API calls. The level of detail for altering the physical level can be of great use for experimenting with padding and frame structure. CNet has limitations such as the lack of wireless protocols and no group addressing or multicast.

3.6 *WebLan Designer*

The WebLan-Designer [84], developed at Auckland University of Technology, uses a web based front end to allow students to design local area networks. The tool does not generate results from a simulator but rather shows the different ways in which a described scenario can be connected, as seen in Figure 9.

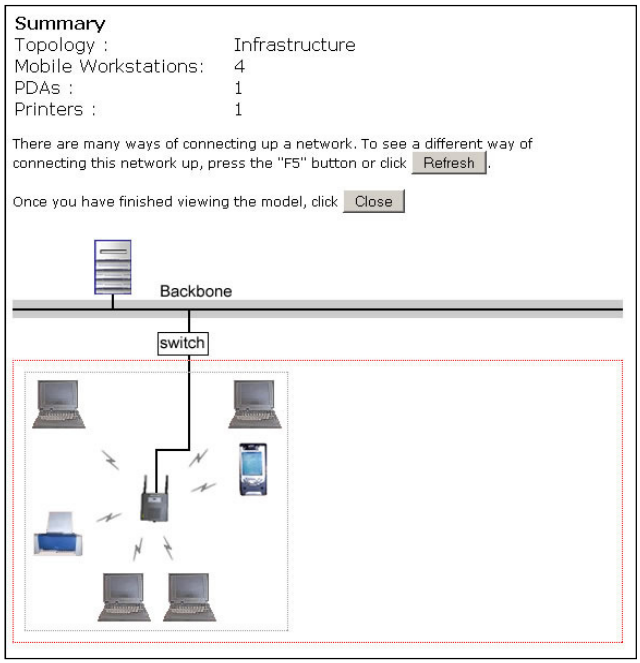


Figure 9: Screenshot of the WebLan Designer [65]

The tool is free to use and is deployed using a standard web page. Both wired and wireless technologies are used. Due to high levels of abstraction, this is a good tool for showing students how to structure a network but restricts deep exploration.

3.7 *ReMLab*

ReMLab [85] is a measurement laboratory which supports remote learning for engineering students. The laboratory is designed to allow remote access through a web page to technical instruments. Students' parameters are entered on a web page which, through Servlet technology, invokes a physical instrument. The physical board is accessed via the Java Native Interface (JNI). The instruments available for experimentation include a digital I/O board, analogue-to-digital and digital-to-analogue conversion through RS232. Concurrent experiment requests are handled using a First in First Out (FIFO) queuing algorithm. Some results require graphical reporting and this is accomplished using a client-side Java applet that generates the graphs. This client-side generation alleviates the server load.

3.8 *SENSASIM*

SENSASIM [86] is a detailed event-based simulator for wireless networks written in Java and developed at the Athens Institute of Technology. The tool is used to teach students about wireless sensor networks, enabling collaborative configuration and viewing of animations. SENSASIM allows students to experiment with different power control schemes, node movement, routing, MAC protocols and the energy constraints of nodes. Configuration is performed through a Java applet and the request sent to a simulation server. The results are then streamed back to the applet where the student can

review the simulation. SENSASIM has been provided using the GNU Lesser Public License [87] and the source code is available to download from the site.

3.9 *iNetwork*

iNetwork is a tool for teaching networks to students. iNetwork allows the construction and testing of wired networks. This is done through an interactive point and click GUI. The software is written in C# and a platform with Win 2000/XP with the .NET framework installed is required to run the program. There are four key components that interoperate to provide functionality. A GUI is used for adding and subtracting network components. The network simulator performs the various networking protocols such as ping, tracert and ARP. A device simulator provides the behaviour of the network components such as switches, workstations and servers. The final component is the network calculator that calculates correct subnets, ARP tables and routing entries. iNetwork provides an experimentation software solution. This is especially useful in institutions where real networking components are unavailable to the students.

3.10 *IREEL*

IREEL [88] is a virtual laboratory enabling students to experiment with networking protocols using a web interface to control experiments over an emulated network. Users sign up at the website [89] and are then able to choose an experiment to run. Example experiments include ping, Iperf [90] and streaming video. Once submitted, a list of experiments and their state is shown. An experiment is pending until the system has carried out the experiment. Once completed, an email is sent out with the results in a range of formats.

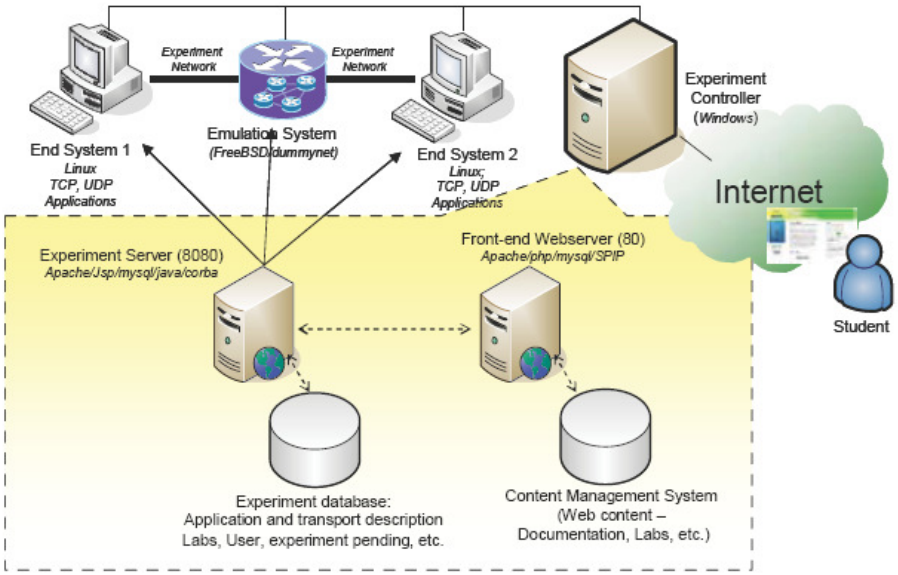


Figure 10: Architecture of IREEL [88]

The user logon is controlled using Shibboleth web Single Sign-On [91] technology, allowing students from participating institutions to log on using their institutions credentials. Once logged on users define their experiment, as shown in Figure 10, using a web front end. The setup is conveyed to the end systems that execute the experiment using real protocol stacks. The emulation system is a single host with multiple interface cards that has a high-level API used for configuring the set up. Packet interference is achieved using FreeBSD/Dummynet [92]. There are three emulation scenarios: static impairments (constant values), timed scenarios or other emulator scenarios. When the hardware is available, the experiment is performed, the result stored in a repository and an email is sent to the experimenter when the results are available for collection.

New experiments are designed and run by course coordinators. The design process involves preparing the results presentation, lecture material and developing a Java controller. The level of user manipulation of the environment is limited. Another problem encountered is the long period of delays between submitting a new experiment and receiving the results, this was in the order of hours. This makes the tool less useful for a predefined 2 hour lab session where students require immediate feedback.

3.11 *MASSIVE*

MAnet Server Suite Incorporating Virtual Environments [93] (MASSIVE) is an emulation environment for mobile ad-hoc networks. The aim of the project is to provide interactive manipulation of an emulated environment, visualisation of topology, management tools and ease of transition of protocols from the emulated to real domain. Network emulations are described as belonging to three domains:

1. Central Control Emulators: Where every node in the emulation is connected to a central server. The central server decides what to do with each packet going through the network. The decision on whether to drop, delay or scramble a packet is achieved via internal parameters. This centralised approach introduces the problem of a single point of failure and creates a significant bottleneck in the system.
2. Simulator Combined Emulators: Simulators such as ns-2 have a hybrid ability to take real world network data and pass it into the simulated environment and back out again. Using this approach only allows the evaluation of top-level protocols.
3. Distributed Emulators: Each node involved makes the decision about how it controls the packet flow. This mode is usually controlled and configured by an external node. An example of such an approach is found in MobiEmu [94] and it this approach which MASSIVE follows.

Each node in MASSIVE contains a server; each server is capable of emulating at least one Virtual Mobile Ad hoc NETwork (VMANET). Each server may instantiate a virtual mobile node and can communicate with other servers. Access and control is accomplished using remote procedure calls (RPC). Route management is performed using iptables [95]. The user can also specify external routing mechanisms.

A user interface is provided that shows information on the network status, neighbours of nodes and route information. A management tool allows the graphical construction of VMANETs as well as control over the MASSIVE servers and VMANETS settings. Configuration is achieved by firstly scanning the network to find available nodes. Mobility of the nodes is done using a scripting language that is embedded in the emulator. The capability for mobility allows the researchers to study the formation of hot spot areas and the implications for certain protocols. The GUI can also be used to create multi-hop routing networks.

Whilst the introductory paper to MASSIVE hints at the use for education, no examples, evaluation or ideas are presented and no further publications have occurred based on this system. The system has also not been made freely available for code review or external testing. While the approach is interesting, the configuration seems difficult and unsupported.

3.12 *Graphical Network Simulator 3*

The Graphical Network Simulator 3 (GNS3) [96] is a network topology simulator containing an IOS [97] emulator [98] to run binary images from Cisco. GNS3 ties together existing projects in Cisco emulation and Pemu, a PIX emulator. In order to build and run simulations in this environment an IOS must be acquired from Cisco. The IOS is a software image that is run on a hardware device and controls access to the hardware components and program logic. The lab tools are aimed at those with advanced knowledge of routing and for those considering the Cisco certification. The user interface is

a GUI, enabling easy addition of various network components. Further configuration is required through Dynagen [99], a text based front-end for using Dynampis.

GNS3 can be run on Mac OS X, Linux or Windows but does require complex configuration and is mainly aimed at advanced users experienced with networking, routing protocols and structures. Another limitation is that running the binary images requires the purchasing of license keys from Cisco which may be prohibitively expensive.

3.13 *PlanetLab*

PlanetLab [72] provides access to a network of computers distributed across the world enabling researchers to conduct experiments. A wide range of research is carried out through this network e.g. content distribution networks [100-102], Internet Protocol Television [103], distributed hash tracking [104] and anonymous communication systems [105]. Membership of PlanetLab is limited to institutions in order to avoid mass abuse and to ensure accountability. Members can create services that may encompass any of these research areas; the individual program running on a single node is referred to as a *capsule*. A *slice* is a horizontal cut of PlanetLab nodes and can be thought of as a network of virtual machines[2] with local resources. The local resources available to a node are defined as a *sliver*. The *capsule* executes its program in a virtual machine with access to local resources. There may be multiple virtual machines running on a single PlanetLab node and a Virtual Machine Manager allocates the resources. Resource allocation is performed using credentials that are issued by a node which specifies the allowed bandwidth and memory usage. Tickets are redeemed at the node and often have timeouts associated with them.

The process of acquiring a *slice* on a PlanetLab node [106] is achieved through three key pieces of information. The first is the service declaring what resources it requires in a statement. The availability to satisfy such a request at each node is then checked through agents requesting status reports from the nodes. The final piece of information given is the number of resources the user is allowed to request and whether or not this matches the service statement.

The TCP Live learning resource [107] enables students to explore the behaviour of TCP on the global Internet using PlanetLab nodes. This enables observation in a wide variety of conditions that the protocol has to cope with, thereby extending their viewpoint out-with the limited scope of their own institutional firewalls. The user interface allowing students to select nodes for TCP traffic to flow between is shown in Figure 11.

[2] A virtual machine is a machine that has access to virtual components and is fully isolated from its host. The host mediates the virtual machine's access to its components.

Figure 11: Distribution of the Planet Lab service TCP Live around the world

3.14 *jFAST*

jFAST [108] is a tool designed to support the learning of finite state machines. The tool supports Deterministic Finite Automata, Non-deterministic Finite Automata, Pushdown Automata, Turing Machine and State Machines. The students construct their formal languages using XML, which allows easy extensibility and transportation between students and teachers. In their evaluation 77% of students expressed dissatisfaction with their knowledge of finite machines and 100% said jFAST would have helped them [108]. This type of evaluation is very different to the evaluation of impact on the students who have actually used the software, of which no information was provided.

3.15 *Virtual Distributed Ethernet*

Virtual Distributed Ethernet (VDE) [109] is an environment which provides the emulation of Ethernet and services that run over it. The system allows the student to alter components of the Internet structure from the routers, switches and the address resolution protocols to the virtual memory hypervisor. There have been difficulties with the large scope of the project; students complained that such realism was accompanied with poor usability [109].

3.16 *Virtual Network System*

The Virtual Network System [110] (VNS) allows students hands-on experience in deploying routers, firewalls and network services that will route real traffic but without the legal complications of traffic snooping on a University network. A VNS server daemon can configure multiple virtual network topologies. Clients can request the traffic seen by any host on the virtual network; the client can then discard, inspect, modify or inject new packets to/from the host. In Figure 12 the client has configured a web server to be located inside a virtual topology, which is hosted by the VNS server daemon. The

client can then run a web browsing session across the Internet, accessing the virtual web server. The VNS client can modify/discard/inject any packets that are destined or originate from the web server.

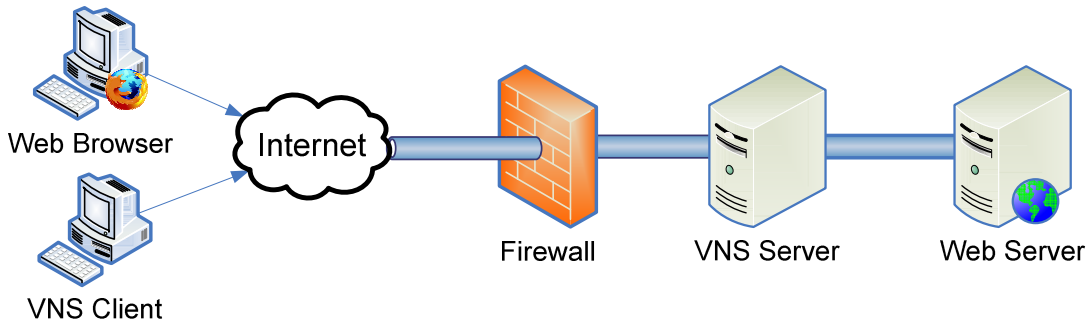


Figure 12: Locations of the VNS server and client during an experiment

A GUI has also been developed for the system, Clack, which is a graphical router builder. This is meant to demonstrate various networking concepts, IP TTL, IP forwarding and slow start algorithm with the signature saw-tooth graphing capability. The feedback from students was described as mostly positive but little in the way of formal evaluation has been provided. Many students stated that there is a lot of work in implementing the projects, as around 1000-2000 lines of code were required. This level of skill is not within the capability of lower performing networking students. However, the tool is aimed at academically strong undergraduates and postgraduates. No examples of the structure of the code or nature of the library are available.

3.17 *Example Deployment of Learning Environments*

In [111] a strategy for teaching ATM and wireless networks with deployed networks to fourth year undergraduate and two graduate courses is discussed in detail. The students were set a series of exercises to complete at differing levels of the OSI stack. The exercises included a data-link layer project aimed at implementing stop-and-wait ARQ and a MAC level comparison performance between Ethernet and ATM. The laboratory consisted of five Intel Pentium computers, a Fore ATM switch and a Lucent WavePOINT wireless bridge. Each computer had an Ethernet, Lucent WaveLAN and Multi-Mode-Fiber ATM interfaces provided. The operating systems supplied were a Windows NT server, Linux server and three dual-boot Linux/NT. Feedback from the experiment was gathered in an informal manner. An unspecified questionnaire was filled in per exercise, mid term and at the end of the course. The feedback provided in the paper was anecdotal, with selected student quotations from questionnaires. The quotes provided indicated a level of engagement from the students. Overall the experience for the students and teachers appears to have been positive, though impact measurement has not been addressed.

In [112] the authors show the impact of using an emulated laboratory in teaching data communications and networking projects. Their approach has been to use OPNET with Wireshark [113] in setting laboratory exercises for students in the fourth year of an undergraduate course. The exercises were to

investigate the TCP receiver windows size. OPNET provides the communication devices and then Wireshark is used to analyse the results. The student learning outcomes were to:

- identify the objectives of the study
- develop/design the simulated model of the system
- setup and configure the simulation
- execute the simulation and collect data
- analyse and validate the collected results

These outcomes are taken from the book by J Banks [114] and detail the rigorous scientific process that the students should be learning to apply when constructing, executing and understanding a simulation. One demonstrator noted that: *"Students seem more attentive when we demo a live trace than when we discuss a static performance graph from the textbook, even when the ultimate result of the trace is essentially the same graph."*

However, problems were encountered, and it was noted that when constructing scenarios, *"Students often have a hard time learning how to configure applications and user profiles, how to deploy applications and how to interpret collected results."*

A more formal approach to feedback was also taken by having students fill in the Student Instructional Report II surveys from ETS [115]. The results showed an increase in student learning and interest in the courses. They conclude that the introduction of such immersive exercises was helpful in their course and will continue to expand its deployment in their curriculum.

Another approach to abstracting the difficult problem of experimenting with networks without having to learn new languages is the Services and Systems Framework [116] (SESF). The projects were a client/server architecture involving data transfer protocol, congestion control and connection management. The evaluation provided was a measurement of dropout rates, number of emails per students and percentage of students who completed their projects. All these metrics showed an increase in engagement. However this does not prove that the tool helped improve understanding of the subject matter.

3.18 *Education through Virtual Worlds*

In [117] a 3-dimensional virtual environment is used to give a more engaging representation of a Uninhabited Aerial Vehicle (UAV) in flight to its pilot. A UAV is where no human pilots are onboard and the mission data is sent from a control centre on the ground. This paper focused on providing a 3-dimensional representation of the state of the plane. Included in this representation are satellite images of the area, 3-dimensional buildings and meteorological conditions. The motivation for such a system was cited as a 17% rate of human error in UAV crashes. The virtual display was appreciated due to the level of realism provided enabling immediate recognition and intuitive understanding of the vehicle's

state at any point. This paper shows the value in creating an intuitive interface that can be easily understood and comparable to real life and shows some perceived educational benefit.

Future of Air Force Education and Training is a whitepaper released by the United States Air Force (USAF) describing "Air Force 2.0" that could be used as a virtual delivery platform known as MyBase [118]. This will provide an exploratory environment for learning both for recruits and civilians. MyBase is being developed to support both continuous and precision learning in an interactive exploratory environment.

The authors of [119] described a system where children were provided with a range of computer games after school. The work aimed to measure learning and cooperation amongst the students. The interactions were analysed using Activity Theory [120] and the Zone of Proximal Development [50]. The data from the experiments indicated a success and found several interesting results. Conflict can be very beneficial when using a Computer Supported Collaborative Learning environment. The CSCL should enable conditions for constructive conflict resolution. The range of games provided included, Sim City [121] and Choplifter [122] to represent simulations in which practice logical and mathematical concepts.

The engagement achieved through the use of a MUVE can provide a strong motivation and interest in the topic area as shown in [123], where a group of students are set a task in a virtual world. The students showed a strong willingness to experiment, construct, implement and evaluate a hypothesis all in a virtual context. There has been other research which supports this engagement through the use of a MUVE [124-125]. Through taped interviews, students described increased motivation and connection in their learning to the real world. An example of a MUVE is Second Life, a detailed description of which is provided in section 4.11. Several universities are involved in Second Life and have purchased islands for educational and recruitment purposes, notably The Open University, University College Dublin, Stanford, Edinburgh University and Harvard. The International Society for Technology in Education (ISTE) [126] has held talks, seminars and presentations *in-world*. Other examples of educational uses in Second Life include foreign language tuition set up by the British Council [127].

National Oceanic and Atmospheric Administration (NOAA) and the Earth System Research Laboratory (ESRL) have created a presence in Second Life [128]. The two agencies have created a simulator called Meteroa that contains interactive educational demonstrations such as a sea life submarine ride, two tsunami demos, melting glacier, a real-time temperature map powered by Yahoo! and an airplane ride into a hurricane. NASA has also created a presence in Second Life [129] producing various educational tools for use by visiting *avatar*s. A MUVE has been used to provide an interface for a museum [130] for middle school students.

In [131], Second Life is used to provide a virtual environment for training doctors. The example scenario used is a virtual patient suffering a post-partum haemorrhage. Students can use the *in-world* voice, virtual medical equipment such as oxygen masks and an IV to treat the haemorrhage. There is

ongoing research into the effectiveness of using Second Life as an educational tool for simulations at Imperial College London. David Taylor stated:

"We tested [the virtual O.R.] in a controlled experiment on 40 first-year medical students prior to their first visit to a real O.R. We wanted to determine if [the SL program] gives them more confidence before their first exposure to the real thing. We've found it is just as effective as the training O.R. in the physical world."

Surveys have indicated that the students' reactions in Second Life are similar to that in the real world and that the simulation has helped. Another benefit of the simulation is exposing students to scenarios that are impossible or unsafe to replicate. The drive from the students is towards an interactive form of simulation and away from the more traditional CD-ROM approach of fixed observation of a scenario.

Another use of Multi-User virtual environment is to study students' approach to problem solving. In [132] students have *avatar*s that interact to solve a simulated 19th century city's problems with illness. Environmental simulation has been attempted using MUVE with focus on the architecture needed for an educational MUVE. The paper [133] provided some architectural details but little on development or evaluation and it appears that work has ceased on the project.

In [134] a researcher at Drexel University created Drexel Island [135] in Second Life for the exposition of chemistry concepts. The island was used to display 3-dimensional molecules, reaction mechanisms and chemistry quizzes. The construction of the molecule in Second Life, allows users to fly around and obtain any viewpoint they wish. This is not the main strength of Second Life as the technology to view and manipulate a 3-dimensional model is well established and available through the use of plugins in current web browsers. What is provided is a series of informational posters explaining the various molecules and animations of docking. Interactive quizzes are provided on Drexel Island for students to work through, with each correct answer progressing them towards the final prize of a molecule. A molecule creator is available which, when the user enters a chemical formula, builds up the molecular representation and displays it to the user. This was achieved through the use of ChemSpider [136] which is a database that provides the structure of molecules in a standard format. The database provides an interface through web services. The developers then wrote a Second Life script capable of creating the molecule from the standard format. The environment also contains pre-scripted animations of molecules interacting which are controlled through simple chat commands. An interactive absorption spectrum is used to show the effect of light on hydrogen atoms. When the light source is turned on, the electrons enter an excited state and release visible light which is observable on the screen behind the experiment. A 3-dimensional periodic table is freely available which allows students to click on the various elements and receive information on each one. The relative size of each element in Second Life is representative of their atomic weight.

MUVEs can also be used to provide realistic environments for fieldwork; such an approach is useful when financial and time constraints restrict the ability to perform real fieldwork. An example of such an implementation is the Laconia Acropolis Virtual Archaeology (LAVA) project [137] based on

fieldwork undertaken at the British School in Athens during 2000/1. Archaeological fieldwork through excavation is often difficult due to lack of funds, inclement weather or damage to the area of interest [138]. LAVA aims to compliment and enhance real-world fieldwork by enabling students to gain experience with the methodology of preparing and excavating a site collaboratively. To do this a realistic representation of the Sparta Acropolis Basilica in Greece was designed firstly in a 3-dimensional engine Jake2 [139]. Jake2 is a Java port of the Open GL release of the Quake II game engine and provides the ability to read map files created using level editors for Quake. Because Jake2 is written in Java it was possible to launch the Acropolis environment through the Java Native Launching Protocol (JNLP) [140] on a modern browser where the Java Runtime Environment was installed.

The Acropolis was later ported into Second Life and Open Simulator with a visitor centre showing relics and information about the site. Students were able to walk around the Acropolis, which was built to scale. This was shown to engage students and feedback proved it to be popular with several classes when integrated into the course curriculum.

The works highlighted here demonstrate that there is a real interest and appreciation of the collaborative and immersive nature that is provided through a MUVE. While Second Life is currently the MUVE of choice, it is mainly due to its widespread adoption rather than any inherent capability over other MUVEs.

3.19 *Miscellaneous Related Work*

In [141] the authors attempt to create a virtual laboratory using the non-declarative modelling language Modelica [142]. The paper proposes a standard modelling methodology through Modelica and demonstrates it by building a virtual laboratory called “VirtualLabBuilder”. The VirtualLabBuilder was used to connect graphical elements to model a water tank. The parameters considered in the laboratory are the pump voltage, gravitational acceleration and an outlet hole.

Exploratory learning has also been shown to be effective in simulating high-performance processors, a system component many students struggle to understand. A user can access the simulator using a web browser and the results are shown visually. The user can construct a processor by supplying a configuration file with the modules listed and their specified interconnections. This extensibility is a very useful feature that allows the tool to be continually updated by anyone. To keep the presentation of information to the user simple, the internal state is hidden but can be shown in a separate frame by clicking on the register/memory location. An example screenshot is provided in Figure 13. The user can monitor these values as they are updated on a cyclic basis as the simulation proceeds. A further example of this can be seen in DLSim 3 which is a multi-level logic simulator demonstrated in [143].

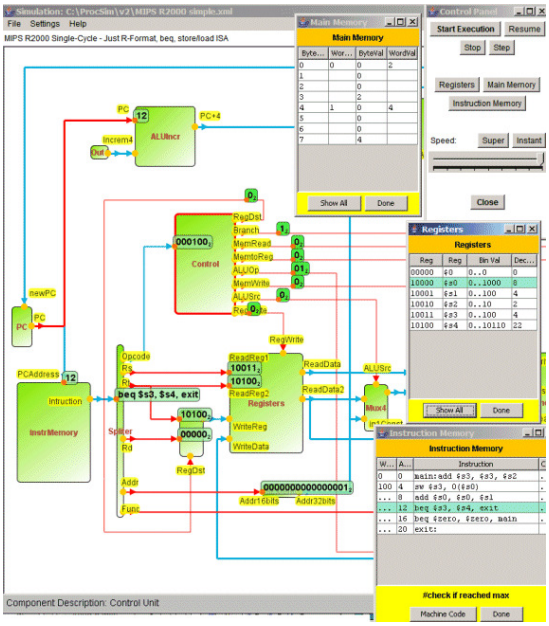


Figure 13: Screenshot of a simulator to enable students to learn how a CPU works

A web-based pico-controller [144] training simulator has been designed at the National University of Science and Technology (NUST) in Zimbabwe. The students scored the tool highly. The results were 4.44/5.00 for the simulator's usefulness, 4.74/5.00 for relevance and 4.96/5.00 for ease of use. The evaluation provided adds weight to the idea of a simulator embedded into a web page, allowing students to "rapidly evaluate functions and interfaces of the electronic device, thereby giving them confidence in approaching the real hardware". Cryptography has also seen an exploratory learning approach [145-146].

The use of graphical representations has been proven to improve users learning over the traditional approach of text [147-149]. The field of using computers for learning resources is expanding with new journals being created and the current Journal on Educational Resources in Computing undergoing a revival. The IEEE now issues a journal named Transactions on Learning Technologies.

3.19.1 PLATO

Computers have been used to provide a Virtual Learning Environment (VLE) for many years. A VLE is a system that facilitates learning in a certain field through specially configured environments. An early and robust VLE for learning is PLATO (Programmed Logic for Automated Teaching Operations) [150] which was developed at the University of Illinois in the early 1960s. This is one of the earliest computer-based VLE and was used to teach students in five key areas at the university (accounting, biology, chemistry, English and mathematics). The system saw continuous development and redesign. A report was published detailing its progress, evaluation and analysis techniques [151]. Subsequent versions of PLATO were designed to cope with 5,000 students using the machine at various distances

from the framework. A series of lessons were created using the TUTOR [152] language in each of the targeted subjects. PLATO usage increased across all subjects as more lessons were created.

A thorough analysis of PLATO is provided in [151]. Firstly, treatment classes were divided into those where the PLATO system was employed and those where it was not. Keystrokes and measurements of time usage were also collected from the main system. This typifies the passive approach taken here by the evaluators, as they wanted a responsive and non-reactive method of analysis. The evaluation was designed so that the curriculum of each subject was collected, lessons on sections were agreed, then pre-tests and post-tests were constructed. The statistical analysis showed no overall improvement in student performance. There were many possible contributing factors to a lack of improvement such as the ability of the instructors and validity of the pre/post-testing. An encouraging result was the improvement of student attitude towards the topic and system. 70 - 90% continued using the system after class, citing they liked being able to make a mistake without feeling embarrassed and found comments from PLATO useful. 97% of instructors found the system to be moderately or very helpful and 80 - 83% judged PLATO to have a positive impact on student attitude and achievement in the topic area. In the final semester, the schools continued to work together on the PLATO system, assigning new courses and lessons. At the end of the evaluation and initial funded, period the school continued to provide funding.

3.19.2 Grid Architectures for E-Learning

The grid provides a system for linking existing Service Oriented Architectures together with the maintenance of state in a users' session. This section will discuss the example usage of the GRID as an e-learning platform and the relevant frameworks and tools used.

SELF

SELF is a Semantic grid-based E-Learning Framework [153] for a comprehensive E-Learning environment that will provide full control over class scheduling, courseware maintenance, user collaboration etc. The approach taken by SELF is not to rewrite every level but rather to tie together existing tools. The authors state that a solution cannot be achieved with a new implementation of every level of the architecture, as this may result in a wasteful rework. Equally, a solution cannot be found with loose integration of existing components as this would lead to an unrealistic and non-scalable framework. As such, they propose that a good understanding of the related works and a good mapping between the two can grow a better architecture.

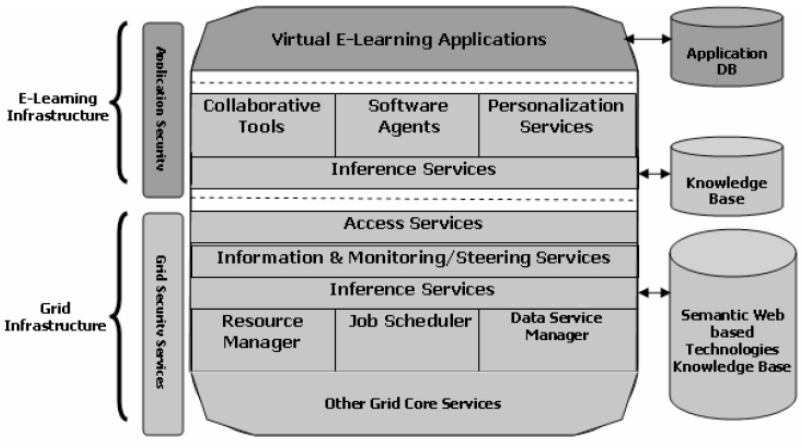


Figure 14: Structure of SELF [153]

Figure 14 shows that SELF attempts a modular approach with existing technologies to provide collaborative tools.

The SELF components include Chandler [154] which is released under the Apache License. Version 2.0 can be used for note taking, email, calendar and tasks. The software is written in Python and has yet to reach 1.0 status. The advantages are a cross-platform, intuitive user interface and a persistent object database. Chandler is also built using defined open standards

CoAKTing

CoAKTinG [155] (Collaborative Advanced Knowledge Technologies in the GRID) started in June 2002. This project aimed to provide technologies to enable enhanced interactions between people. Edinburgh University provided task-oriented messaging, planning and activity management. KMi, the Open University, worked on online presence, availability, real time conversational mappings of meetings and group memory capture using Compendium [156].

Grid-Based SENSASIM VSE Simulator for Sensor Networks

Continuing the work described in section 3.8, SENSASIM has been expanded to enable service oriented access through the grid [157]. The SENSASIM grid-based VSE simulator was deployed in Athens Information Technology and the CyLab in Kobe, Japan. Encouraging results were reported after two test deployments.

3.20 *Summary*

This section has provided a review of selected applications that have been used for exploratory learning. Table 6 compares the features of each of the selected applications. WiFiVL I, WiFiVL II, WiFiSL and WiFiOS all make use of a network simulator to provide an animation of a scenario to the student. Unlike IREEL, iNetwork, GNS3 and MANET, the WiFiVL framework does not make use of emulation. The WiFiVL framework described in this dissertation makes use of a complex simulator and can be extended to include emulation results. Like all the networking related works listed, it does

attempt to convey a networking protocol to the user in an intuitive interface. Non-networking related works show other approaches to engaging the student and in conveying a topic or idea. There have been several attempts to create immersive virtual environments for teaching diverse topics but none that have focused on wireless networking.

4 Technologies and Concepts

This chapter describes the programming languages, systems and concepts used in the development of the WiFi Virtual Laboratory (WiFiVL). Where appropriate, examples are provided to illustrate a technology.

The IEEE 802.11 [158] protocol is one of the most pervasive wireless protocols. This is reflected in the large number of "hot-spots" available in the UK [159]. Section 4.10 details aspects of the protocol that are exposed through the WiFiVL.

Each of the technologies described in this chapter has been used in some combination in WiFiVL I, WiFiVL II, WiFiSL (WiFi Second Life) and WiFiOS (WiFi Open Simulator). Table 7 summarises the area of application for each of the technologies discussed in this chapter.

Type	Technology	WiFiVL I	WiFiVL II	WiFiSL & WiFiOS
Simulator	**ns-2**	✓	✓	✓
Servlet Container	**Apache Tomcat**	✓	✓	✓
Programming Language	**ActionScript**	✓	✓	
Programming Language	**Java**	✓	✓	✓
Programming Language	**JavaScript**	✓		
Document Format	**XML**	✓	✓	
Multimedia	**Flash**	✓		
Multimedia	**RIA**		✓	
Virtual World	**MUVE**			✓

Table 7: A comparison of technologies used in WiFiVL I, WiFiVL II, WiFiSL and WiFiOS

4.1 *Types of Simulators*

Simulators are forming an increasingly large role in many fields. In Biology, simulators are used for protein folding [160] in attempts to cure cancers. The defence industry have used simulators for battlefield tactics [161] and the modelling of entire socio-political countries [162]. There are several types of simulators. This section discusses discrete event, Monte Carlo and continuous event simulation.

The Monte Carlo [163] technique, coined by Ulam, involves randomness. A range of random inputs are generated; these are analysed using a deterministic model and the output recorded. The statistical

analysis of the output provides the approximate answer. As the number of inputs increase, so does the accuracy of the output. This method is useful whenever the possible range of inputs is too large to be performed systematically. This technique saw great use with nuclear experimentation but is also now used in financial and telecommunication systems evaluation as well as physical sciences.

A continuous event simulation is used for biological and social systems that can be described using differential equations [164]. The equations are estimated and then periodically solved to produce a feedback loop that can then affect the state of the simulation.

Discrete event simulation performs a series of events at specified intervals. This technique of simulation maintains a monotonic clock and events occur at a discrete time. A list is maintained of pending events to be simulated. Each event in the simulation may affect the finite state machine, causing a range of other events to occur. For example, a Request To Send packet in ns-2 will generate a Clear To Send packet; these events occur at discrete time values and affect each node's finite state machine. The output is a trace of all events that have occurred during the simulation and the system time at which they occurred. Examples of this type of simulator are ns-2 [5] and Avrora [165].

4.1.1 ns-2 the Network Simulator

The Network Simulator 2 (ns-2) [5] is a discrete event network simulator which is heavily used in research for modelling and evaluating protocols. A Linux environment is required for ns-2 as well as the Tool Command Language (TCL) [166] and Tk framework [167]. Creating new simulations requires programming knowledge of OTcl [22]. ns-2 itself is open source and has many contributors to its source code including Defence Advanced Research Projects Agency [168], Palo Alto Research Center [169] and University of California Berkeley [170]. Contributed code includes many areas such as routing, wireless, satellite, topology, transport and application. The contributed code for wireless simulations allows improved Medium Access Control (MAC) and physical layer modelling. There is a large amount of open source activity relating to the wireless code and this reflects the amount of interest and the importance in teaching this wireless technology. Contributions provide source code for new standards (e.g. IEEE 802.15.4 [171]), revisions of existing ones and patches for the system.

ns-2 is developed using the language C++ [172] due to its speed. A disadvantage is that changes made to any C++ file require a recompilation of the whole project. This disadvantage is countered by using OTcl. OTcl allows interactions between objects created in C++ and OTcl. This gives ns-2 the flexibility to allow scripting of complex scenarios in OTcl without having to recompile for every new scenario. Drawbacks to this approach are the requirement to either run predefined scenarios or having to learn OTcl, and the requirement of a Linux environment. Using the official API to program the OTcl code has proven to be difficult and often cryptic. Several methods that are specified in the API do not exist in the OTcl space. Other OTcl scripts compile without error but then cause an irrecoverable termination of the simulation. Tracking down such bugs is both time consuming and difficult. Network Animator (NAM) is provided with the ns-2 release. NAM requires X-Windows

libraries to run and is written in TCL/TK. It is used to animate trace files generated from an ns-2 simulation. The performance of NAM is limited as it has a variety of bugs and can only support a limited range of animations.

An ns-2 simulation has an initialisation [173] step which creates the packet format, a scheduler and an agent. The scheduler selects and executes the next time ordered event. The simulator is single threaded only. A topology needs to be created in ns-2 and this is done using the primitive *ns*. The node is a standalone OTcl class and consists of two Tcl Objects, an address and a port classifier. All nodes contain an id, (which is monotonically increased from 0) a list of neighbours, agents, a node type and a routing module.

Nodes can be configured using the *node-config* setting in OTcl. An example of how the node is configured is shown in Figure 15. In this example the type of MAC protocol, queue type, antenna type, routing protocols and components that should be traced for review are configured.

```
$ns node-config -adhocRouting DSR \
-llType LL \
-macType Mac/802_11 \
-ifqType CMUPriQueue \
-ifqLen 32768 \
-antType Antenna/OmniAntenna  \
-propType Propagation/Shadowing  \
-phyType  Phy/WirelessPhy \
-channelType Channel/WirelessChannel \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace ON \
```

Figure 15: OTcl code fragment that defines a range of node properties

When a node receives a packet, the packet's fields are examined and then the values are mapped to an outgoing interface, which is another object. This analysis is done using the classifier object and a table lookup to calculate where to send the packet. The packet is examined and forwarded onto the appropriate agents or outgoing links. Routing modules can also be developed and inserted into ns-2. A routing module can consist of three parts:

- Routing agent – Exchanges a routing packet with neighbours
- Route logic – Uses information gathered by routing agents to perform route computation
- Classifier – Located at the actual node and uses the computed routing table for packet forwarding

Agents are a core concept of ns-2 and are the source of network packets construction and consumption and are created in the OTcl segment. The agents know the source and destination addresses, size and type of packets. There are several Protocol Agents in ns-2 including: TCP, TCP/Reno, TCP/Vegas, UDP, RTP and RTCP. An example of how to define an OTcl agent is shown in Figure 16. In this example a TCP connection is made between two nodes and then an FTP [174] transfer takes place. In this example, the communication starts at time 0 and continues for 0.5 seconds.

```
set protocolDescriptor4 [new Agent/TCP]
$protocolDescriptor4 set class_ 2
set sinkDescriptor6 [new Agent/TCPSink]
$ns attach-agent $node_(1) $protocolDescriptor4
$ns attach-agent $node_(0) $sinkDescriptor6
$ns connect $protocolDescriptor4 $sinkDescriptor6
set applicationDescriptor5 [new Application/FTP]
$applicationDescriptor5 attach-agent $protocolDescriptor4
$ns at 0.0 "$applicationDescriptor5 start"
$ns at 0.5 "$applicationDescriptor5 stop"
```

Figure 16: OTcl code fragment for defining a TCP Agent

Figure 16 demonstrates that OTcl is quite cryptic and presents a barrier to using ns-2 as a learning tool. This is not a criticism of ns-2, which has not been designed as an educational tool.

4.2 *Usability Factors*

When designing a Graphical User Interface (GUI), it is important to take into consideration the field of Human Computer Interaction (HCI) and the usability principles that have been formulated. Developing a GUI with these principles avoids confusing and unintuitive interfaces. This section details some design heuristics that have been developed to help create usable interfaces. This list of heuristics has been elicited from publications in the field of HCI [175-179].

System Status

System status should always be visible with appropriate feedback to the user [13], e.g. a progress bar showing the amount of a file remaining to download. This continuous feedback stops the user from having to speculate if the system has malfunctioned.

Error Prevention

Error prevention is a key heuristic of usability. It should be non-trivial for the user to cause an unrecoverable error. Systems may occasionally crash and it is important that there is an appropriate reporting mechanism. This reporting mechanism should make it easy for the developer to see the problem and provide suitable feedback. The user should be informed why the software has failed and, if it was their responsibility, how it can be fixed. Opaque technical descriptions should not be displayed to the user, e.g. a memory stack trace that requires programming knowledge to understand, as this is unhelpful.

User Memory Load

The memory load of a user can be lessened by having an interface that does not require remembering settings between stages of the system. The user should be helped in carrying out their main task rather than having to remember detailed information to complete their task.

Shortcuts

The system should support shortcuts for advanced users, e.g., enabling the copy mechanism in a text editor with the keyboard command ctrl-c. While it is important to have an interface through which a

user can clearly express their intentions, there should still be a shortcut way to configure certain aspects for advanced users.

Help

If a user finds a certain task unintuitive, they should be able to complete their task through reading the system help documentation. The documentation should be easily searchable and should succinctly explain a solution.

Perceived Affordances

A user interface must provide perceived affordance, that is one should avoid a situation where users do not realise that they need to click on a certain item for some functionality. For example, a violation of this is the drag-and-drop components that are unobvious as to what needs to be dragged or that a section is even drag-and-drop enabled.

Fitt's Law

Fitt's law [180] relates the time taken to move to a target area as a function of the distance and size of the target. Having small targets for a user to click on causes severe usability problems for the elderly and those with poor motor skills.

Undo

Irreversible actions should not be facilitated with one click, e.g. a reset button on a web form that will confirm the user's intentions before clearing user-entered data with a dialogue box.

Default Values

Default values for options should be provided where appropriate. Users, when confronted with an input or choice, should be able to use a system-set default. This allows novice users to speed up some interactions with the interface and provide some idea of the expected value of entry. Providing default values also indicates to the user the type of input required, e.g. whether the expected input is a number or a series of characters.

Enjoyable

Malone [181] describes heuristics for enjoyable user interfaces through the lessons learned from computer games. In this work, key features are highlighted: mastery of the environment should be intrinsically worthwhile, features should be revealed incrementally and the system should be useful without any formal training.

4.3 *Servlet Technology*

Servlet technology [182] is used to enable server-side Java applications to easily interact through a web interface providing dynamic output. A JavaServer Page (JSP) file is a HTML document with annotated sections that incorporate Java code. This JSP file is then compiled into a servlet. A servlet is executed

in a web container, an example of which is Tomcat [183] (see section 4.3.1). The container instantiates a servlet and invokes its *init* (initialise) method. This is performed once at the start of the lifecycle. Each servlet can be used to handle different HTTP methods such as Get and Post. A servlet is removed with the *destroy* method. This method is called once in its lifecycle.

For portability and manageability, large web application projects are compiled into a Web Archive (WAR). A WAR is a compressed archive containing the compiled classes, libraries used and any other resources required such as images and style sheets. A WAR file is deployed to a web container by uploading to a predefined directory. The container will unzip and *deploy* the servlets, invoking the *init* method on each one. To *un-deploy* a WAR, it may be simply deleted, causing the container to remove all unzipped contents and any mapping in its memory.

4.3.1 Apache Tomcat

Apache Tomcat [183] is an open source project by the Apache Software Foundation to implement the Java servlet and Java Server Pages (JSP) [182] technologies. Apache Tomcat contains three main components:

- Catalina is a container that allows the deployment of Java servlet and Java Server Pages (JSP) [182]. Web applications are compiled into a compressed archive with the suffix WAR. Once deployed to Apache Tomcat, the archive is unpacked and initialised. Catalina manages security, concurrency and the life-cycle management.
- Coyote is the HTTP Connector used in Apache Tomcat. Coyote maintains an open socket that listens for connections from a client. When a connection is initiated, it is directed to the Apache Tomcat engine and sends the response back to the client.
- Jasper parses the JSP files and compiles the servlet to be handled by Catalina.

4.4 *Java Beans*

If an object in Java has simple getter and setter methods for a set of fields inside the class with an empty constructor then it is called a Java Bean [184]. An example of a Java Bean is shown in Figure 17. The constructor in Figure 17 accepts no arguments and the values inside the class can be accessed and modified through get and set methods. A Java Bean must be able to transmit its state as a byte array, known as Serializable.

```
public class SimpleBean {

    String valueA;
    String valueB;

    public SimpleBean() {
    }

    public String getValueB() {
        return valueB;
    }

    public void setValueB(String valueB) {
        this.valueB = valueB;
    }

    public String getValueA() {
        return valueA;
    }

    public void setValueA(String valueA) {
        this.valueA = valueA;
    }
}
```

Figure 17: Code excerpt showing a simple Java Bean

The requirement of a constructor without any arguments facilitates the instantiation of the class by other frameworks.

4.5 *XML*

eXtensible Mark-up Language (XML) is the structuring of information in a user-defined way using custom descriptive tags in a text format. XML is the lingua franca of Service Oriented Computing. XML enables meaning and context to be provided to a set of information and is derived from the Standard Generalized Mark-up Language (SGML). SGML is a meta-language used for data representation. SGML evolved into an accepted format for transferring information over the Internet. The World Wide Web Consortium [185] (W3C) created a formal specification of HTML based on SGML. As the limitations of HTML became apparent, XML was formed allowing future expansion from users. XML also enables the processing of information by machines.

An XML document can adhere to pre-defined structures. A defined structure is specified through a Document Type Definition (DTD) document. A DTD document is not written using XML but using its own syntax. DTD has been criticised as it is difficult or impossible to capture the definition of complicated XML documents and there is no support for namespaces. XML Schema Definition (XSD) is mostly used in place of a DTD as it describes an XML document using XML syntax. XSD also has support for namespaces and is more accurate at reflecting structures in an XML document.

eXtensible Stylesheet Language Transformation (XSLT) document enables the representation and modification of XML files. XSLT can display an XML file in a non-XML way with the formatting required; this can be used to display XML files to users appropriately on a webpage. XSLT can also transform the structure of the XML document allowing the transformation from one document to another.

4.6 *Flash with ActionScript*

Flash [186] was unveiled in 1996 to enable animation and multimedia in web pages. The animation is performed through the moving of images and text. This is different from a video where each frame is rendered and essentially static. The scripting language ActionScript 2.0 was introduced with FlashMX [187] as a way of controlling elements on screen. Through the use of ActionScript, elements can be added or removed from a scene and their exact positioning controlled. ActionScript also contained methods for XML processing and manipulation. The language is similar in style and syntax to JavaScript. Flash was originally owned by Macromedia but was subsequently sold to Adobe in 2005 who continue to support its development.

4.7 *JavaScript*

JavaScript [188] is a scripting language used predominantly in web pages to enable client-side dynamic interactions. JavaScript was developed by Brendan Eich whilst working for Netscape in 1995. The scripting language was later standardised by Ecma International [189].

This primarily client-side language allows for input validation and richer user interactions. It has been expanded to include Ajax (Asynchronous JavaScript and XML), enabling the communication between a web page and the server without the traditional POST and GET submissions. JavaScript is a dynamic, weakly typed language so variables can be an integer or a string without explicit declaration of a type. This is the opposite of strongly typed languages like Java as shown in Figure 18.

```
JavaScript:
        var a = "a";
        var b = 2;
        //this will put the value a2 into var c
        var c = a + b;
Java:
        String a = "a";
        Integer b = 2;
        Integer c;
        //invalid as variable a is not converted to an Integer
        c = a+b;
```

Figure 18: Example JavaScript and Java illustrating dynamic typing

JavaScript allows for user created objects through the use of associative arrays. Objects can be described using the function command as shown in Figure 19. An associative array is an abstract data type which contains a unique set of keys which each hold a value.

```
function student(name, age){
 this.studentsName = name;
 this.studentsAge = age;
 this.studentInformation=function(){
        return "Student Name:"+this.studentsName + "\nAge" + this.studentsAge;
        }
}
var a = new student("eg1",21);
print(a.studentInformation());
//example of how JavaScript performs associative arrays for object orientation:
print(a["studentsName"]);
```

Figure 19: JavaScript example code demonstrating custom object creation in JavaScript

4.8 *Rich Internet Application Frameworks*

A Rich Internet Application (RIA) provides a framework for facilitating detailed user interfaces to be easily created by developers. A RIA framework has extensive XML retrieval and processing capabilities. The use of XML for creating more structured programming interfaces in web applications is exemplified by the increase in use of Asynchronous JavaScript and XML (AJAX). Declarative XML allows user interfaces to be described using an XML document through the use of recognisable tags to describe structure and functionality. The XML document has a schema which can be used to verify a working interface has been described. Because the schema is made publicly available it provides developers with a wide choice of Integrated Development Environments (IDE) to use.

There are different frameworks available for developing and deploying a RIA. Table 8 provides a comparison of three main RIA frameworks. An important difference is Microsoft's decision not to use an XML-derived language for declarative interface construction. Sun Microsystems were late in releasing a RIA framework to the community and as such Adobe's Flex had gained significant developer adoption and prevalence in website deployment. Adobe provided an extensive Integrated Development Environment (IDE) which allowed programmers to design, implement and deploy a RIA quickly and easily with the resulting user interface viewable in all modern browsers.

Name	JavaFX	SilverLight	Flex
Developer	Sun Microsystems	Microsoft	Adobe
Release Date	December 2008	April 2007	June 2007
Development Environment	Netbeans plugin	Microsoft Visual Studio	Eclipse plugin
Programming Language	Java	Any .NET language	ActionScript
Operating System	Java Runtime Environment	Microsoft Windows	Adobe Integrated Runtime
XML Declarative Interface	×	✓ (XAML)	✓ (MXML)
Licence	EULA	MS-EULA	Adobe AIR EULA

Table 8: Comparison of three rich internet applications

4.9 *Model View Controller*

The Model-View-Controller (MVC) [190] design principle was developed at Xerox in 1979 and was originally related to the Smalltalk [191] language. The MVC approach splits the design space problem into three parts – model, view and controller. The Model component represents knowledge. A user wishes to interact in some way with this knowledge, so a view is provided of that knowledge to the user. The view is a way of representing or visualising that knowledge. A set of controls are provided to manipulate or interact with that knowledge through the view.

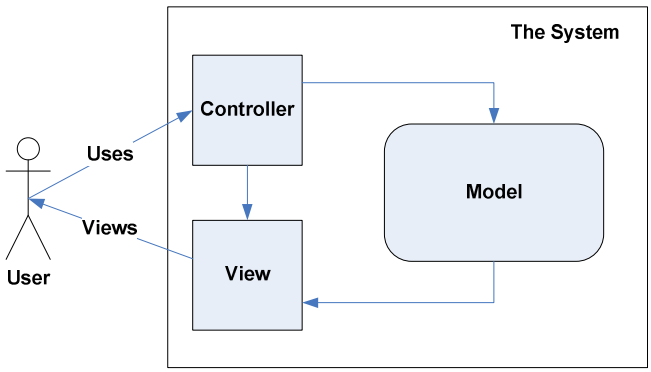


Figure 20: Interactions of the model-view-controller design pattern

The diagram in Figure 20 above shows that the MVC approach should be transparent to the users' experience of the system. The controller may update both the model and the view whilst the view should have no control over the manipulation of the model. The view can act as a "presentation filter" in choosing to ignore certain details present in the model. The controller should never enhance the view by performing actions such as the arrangement of icons in a view. The controller allows for user interaction and will pass the messages from user to one or more of the views.

An alternative approach is the presentation-abstraction-control (PAC) [192] design. The presentation layer defines the behaviour of the user interface input and output. The abstraction component implements the functions that the user interface avails to the user. The control maintains consistency and the overall context between the user and the application. PAC combines input and output into a single component in this system, with one component left to maintain consistency [179].

4.10 *IEEE 802.11 Protocol*

The IEEE 802.11 [158] protocol has widespread adoption in public and private residences. Many devices are capable of communicating using IEEE 802.11 (wireless controlled toys, streaming audio centres, webcams, phones and iPods). IEEE 802.11 networks are prevalent in airports, trains and coffee shops. This large deployment demands that current networking courses ensure that students have a good understanding of this ubiquitous protocol. This section provides an overview of the IEEE 802.11 protocol taken from the standards document [158].

Definitions:

Clear Channel Assessment (CCA) – Logical function performed at the physical (PHY) level that determines the current state of the wireless medium.

Station (STA) – Any device implementing IEEE 802.11 for MAC and PHY interactions.

Basic Service Set (BSS) – A set of stations that have synchronised and are sharing the same coordination function (ad hoc network).

Access Point (AP) –Provides access to a distribution service (this may be done through multiple APs connected via a wired connection).

Distribution Coordination Function (DCF) – Allows STAs to listen to its neighbours to determine if they are currently transmitting and therefore should wait before attempting a transmission (using CSMA/CA).

Network Allocation Vector (NAV) – The NAV is an indicator of when the STA may not transmit on the medium even if it is sensed as being idle.

Point Coordination Function (PCF) – Operates at the AP of network and determines which STA currently has the right to transmit. This is a polling operation primarily performed by the PC. Through the use of beacon frames, it updates the STAs NAV counter and as such can provide contention free access methods.

The topology in which the protocol must perform is dynamic and has asymmetric propagation properties. No assumption can be made about each STA's visibility to another or where the physical boundary of the network may be.

A set of IEEE 802.11 services and their functionality is provided below:

- **Authentication** – Establishes the identity of a given STA and its permissions to associate with other STAs.
- **Association** – Establishes a mapping between a station and access point.
- **Deauthentication** – Removes the recognition of a station as having a given identity.
- **Disassociation** – Removes the mapping between the station and access point.
- **Distribution** – Provides Medium Access Control Service Data Units (MSDU) within a distributed system.
- **Integration** – Provides a link between an IEEE 802.11 and non-IEEE 802.11 network.
- **Reassociation** – Enables an existing station's mapping with an access point to be transferred to a different access point.

4.10.1 MAC Frame Formats

Each frame consists of a MAC header (frame control, duration, address, sequence control information); a variable length frame body and a Frame Check Sequence (FCS) that contains a 32-bit cyclic redundancy code (CRC) [193]. An example of the MAC frame format is shown in Table 9.

Octets:	2	2	6	6	6	2	6	0 – 2312	4
	Frame Control	Duration/ ID	Address 1	Address 2	Address 3	Sequence Control	Address 4	Frame Body	Frame Check Sequence
	← MAC Header →								

Table 9: Example MAC frame format

4.10.2 **Media Access Protocol**

Participating stations (STAs) must have a standard way of using the shared wireless media. The access method used is Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Inter-frame Space (IFS) is used to specify the time interval between certain frames; there are four different types.

1. Short inter-frame space (SIFS) – Time spacing before an Acknowledgement frame (ACK) or polling of a node by PCF. When an STA receives a frame that contains its address, the STA waits the SIFS period then sends an explicit acknowledgement.

2. PIFS PCF inter-frame space – This is the amount of time a media has to be quiet for before any STA can transmit when not in contention mode. This value is shorter than the DIFS which is used in contention mode.

3. DIFS DCF inter-frame space – An STA wishing to transmit a data frame must sense that the media is quiet for the DIFS specified period before transmitting.

4. Extended inter-frame space (EIFS) – If an STA detects that a frame transmission was not correctly received (done via FCS value) then the station waits for the specified time to allow the destination STA to indicate otherwise. If confirmation of a successful frame is received in the EIFS period then EIFS is terminated.

When an STA wants to transmit, it firstly senses the medium and checks if another STA is in the process of transmitting. If the medium is quiet then transmission may begin, otherwise a back off value is observed before sensing the medium again. The back off value is calculated as a random integer in the range of zero to the Contention Window (CW) multiplied by the slot time. The slot time is the time taken to switch the station from receiving to transmitting. The CW is increased exponentially from its original value CW minimum to CW maximum.

Randomness is included in the back off calculation as if the media has been busy for a long period there may be multiple STAs wanting to transmit. If all STAs accessed the media immediately when the medium goes quiet it would cause a large number of collisions.

All directed traffic requires an ACK, otherwise the sender performs a retransmission. The carrier-sense mechanism so far has been a physical method but a virtual one is also included in IEEE 802.11. This is done by advertising to all STAs that the media is about to be used for a certain period of time.

An RTS/CTS exchange is used after CSMA/CA (physical and virtual) has been performed and is used before the transmission of a data frame. An STA wishing to send a data frame to a destination STA sends a Request To Send (RTS) frame (observing previous media access rules as shown in Figure 21). The destination STA, if available to receive data, responds with a Clear To Send (CTS) frame. Both the RTS and the CTS frames contain a Duration/ID field defining the period of time that the media will be in use to transmit the data frame. It is also less costly to send the RTS/CTS frames to ensure a transmission path than waiting for one STA to transmit an entire data packet, not receive the ACK and have to retransmit; also the RTS/CTS frames can work with an overlapping BSS. The RTS/CTS mechanism is used when the data being sent is above a certain value (dot11RTSThreshold attribute),

which may be set on a per-STA basis. Even if a STA has effectively switched off, the RTS threshold attribute it is still required to respond to an RTS addressed to the STA with a CTS frame.

4.10.3 Reassociation

Reassociation allows a mobile node to move between two access points. The services provided in IEEE 802.11 allow a node to move between base stations within an extended service set. A node can receive an MLME-REASSOCIATE request primitive causing the STA to transmit a Reassociation Request frame to the new access point. The new access point can then respond with an MLME-REASSOCIATE confirm primitive indicating a successful reassociation. If a confirmation primitive is not received, then the node has failed to associate with the new access point.

A node can also request a reassociation by sending the reassociation request frame. The access point, on receiving such a frame, will check if the node is authenticated, if not then it will send out a Deauthentication frame. If the access point grants access, a Reassociation Response frame is transmitted with the status code of successful.

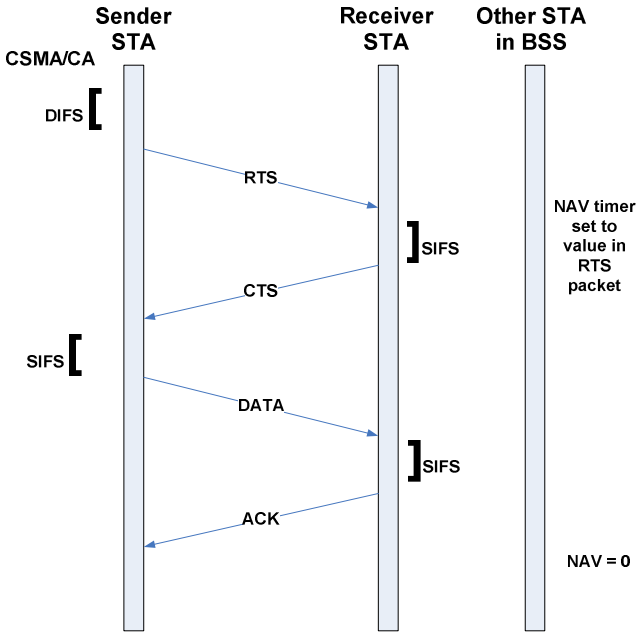


Figure 21: Key interactions on an 802.11 network

4.10.4 Wireless Interference

Interference is caused by two waves of similar frequency causing superposition of the signal. This results in a signal being unintelligible by a receiver at the point where such interference is occurring. The problem of interference is an emerging one in today's society as the 2.4GHz spectrum is being aggressively filled with WiFi, cordless phones, microwaves and other wireless devices and toys. WiFi employs a spread spectrum technique to use different frequencies of the 2.4GHz spectrum, allowing more data to be sent out at once than if the signal used one frequency only. Frequency hopping can

also decrease the chance of interference and increase security. Bluetooth performs frequency hopping at approximately 2000 hops per second. This extreme hopping causes a flood of activity on all the frequencies resulting in lower throughput for other applications.

4.10.5 Hidden and Exposed Terminal Problems

The hidden terminal problem arises in a wireless network when two sending nodes are out of each other's sensing range. Both nodes attempt to send to a third node that both can sense. If this is done at the same time, a collision occurs at the destination node. Before sending, the nodes will perform carrier sensing to see if the media is active. If two nodes are in range of each other and sense the environment at the same time, then neither will sense the other thus both nodes will transmit simultaneously, causing a collision at the destination node.

A solution to the Hidden Terminal problem is to use the control packets "Request To Send" (RTS) and "Clear To Send" (CTS). These control signals request permission from the destination node if it is prepared to receive a wireless signal. The destination node signals its readiness to accept a data packet by sending CTS back to the host. The sender will then despatch the data packet to the destination node occupying the physical media for an amount of time advertised in the RTS packet. The RTS/CTS packets are very small in terms of transmission time and packet size. This dramatically decreases the likelihood of a collision occurring due to identical transmission time from two nodes. RTS/CTS can be used to solve the Hidden Terminal problem, as both nodes would have to request permission to send from the destination host. If node A sends an RTS to node B, it can respond with a CTS that both A and C can hear. C recognises that a transmission will take place and updates its Network Allocation Vector (NAV) timer to maintain radio silence for a given time.

The NAV timer is used by nodes to record the amount of time they expect the media to be busy and is updated using values contained in the RTS/CTS frames.

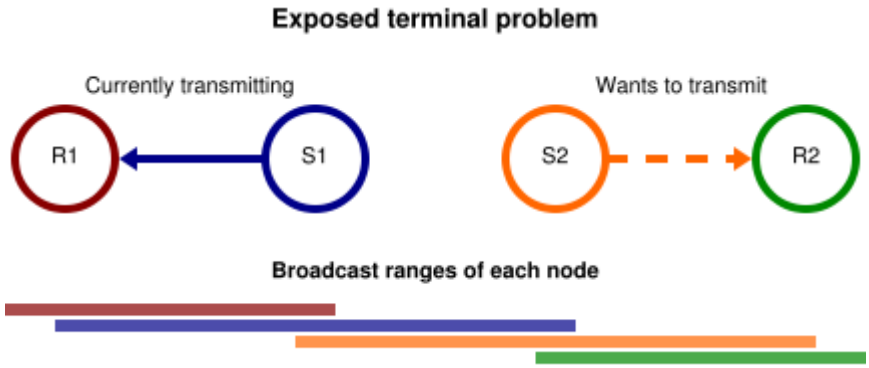


Figure 22: The exposed terminal problem [194]

In the exposed terminal problem, shown in Figure 22, whilst S1 is transmitting S2 is unable to transmit to R2 as it senses the environment is busy. Interestingly, R2 is outside the range of S1's transmission

and as such would experience no interference at that point if S2 were to transmit to R2. This is known as the exposed node problem.

4.11 *Multi User Virtual Environment (MUVE)*

A Multi User Virtual Environment is a representation of a 3-dimensional space that users can interact with using an *avatar*. The interaction capability is provided by using a software client on a computer with suitable hardware support such as a high-end graphics card, processor and memory. The environment, described as an island, can potentially model real life such as land, air and sea with support for the laws of physics such as gravity, momentum etc. Fantasy environments are possible such as abstract existences devoid of form. Each island can have its land subdivided into *parcels* and ownership assigned to an *avatar*. There is support for more than one user to exist in an environment concurrently and mechanisms for users to interact with each other. The representation of users in these environments is through an *avatar*; the word *avatar* is derived from Sanskrit, meaning incarnation. *Avatars* may take any form from abstract shapes to more familiar human or mythological forms. An *avatar's* action has an effect on other users in the environment and is reflected in other viewers. Users may interact with the environment in which they inhabit and any alteration can persist through time. As such, it is possible to create objects, shape the earth and have other *avatar*s see this and equally interact through collaboration.

Each *avatar* has an *inventory* which enables the storage and retrieval of items. This may range from primitive objects such as a cube to groups of primitive objects such as a house. The grouping of objects in an *inventory* allows for objects to have scripts and textures attached and shown to the user as a single item.

A MUVE also supports communication between users through textual methods such as instant messaging or broadcasting text within a certain radius, thus imitating real world conversations amongst groups physically gathered. Voice communication is provided *in-world* for users that have a microphone. Signposts and billboards can be constructed to convey information between users or to indicate an area of land where others can build objects freely. Communication between *avatars* may also be done by the representation of physical actions such as clapping, waving, nodding etc. A form of asynchronous communication is performed using the *Notecard* item. A *Notecard* contains a text document which is often used to convey usage instructions for a parcel of land or an object.

Many MUVEs provide a method for users to programmatically define behaviours and functionality within the environment. An *avatar* can add objects into the world; these objects can be joined with others to form larger structures. Objects may also contain a script which can be event driven. The scripts states can reflect nearby objects, interactions with *avatar*s and many other events. The scripting language used is simple to understand for programmers experienced in any other language. An *in-world* editor is provided to enable script construction and compilation. The scripts are executed in the

virtual world's simulator. Virtual world simulators allow all scripts equal time to run with similar memory resources allocated per script.

This dissertation has subdivided the activity of education in a MUVE can be divided into:

- Collaborative Simulation – This is where several students will collaborate to create a simulation of a certain effect and then review the results.
- Virtual Laboratory – This consists of an environment wherein actions may be performed that result in explainable reactions. These include Chemistry laboratories detailed later in this section.
- Virtual Fieldwork – This allows students to perform fieldwork that would either be too time consuming, costly or impractical to perform in the real world. This may also act as a preliminary training ground before future fieldwork. Disciplines that employ this method include Archaeology, Geology and Sustainable Development.
- Game based learning – This activity is where students, through the use of a game, develop an understanding of a field or concept.
- Lectures – Virtual worlds can be used to deliver lectures or informal talks on subjects and is analogous to students attending a lecture in the real world.

4.11.1 A Comparison of MUVEs, Massively Multiplayer Online Role-Playing Game and First Person Shooters

This section discusses the components used and differences in the genres of MUVE, Massively Multiplayer Online Role-Playing Game (MMORPG) and online First Person Shooter (FPS). Modern 3-dimensional games are constructed of distinct components, many of which are interchangeable. The main components are:

- Physics Engine – A physics engine defines the behaviour of objects in the environment and is generally an implementation of Newtonian physics. Example settings controlled by the engine are gravity, weight and surface friction. Some objects may be exempt, exceptions include the ability of *avatars* to fly.
- 3-Dimensional Application Programmers Interface (API) - Each game that provides a 3-dimensional environment must program to an API to enable the display on graphics cards. The most prevalent interface is Direct3D which is part of a group of APIs provided by Microsoft called DirectX [195]. An alternative API is the Open Graphics Library (OpenGL) [196]. OpenGL provides the programmer with a way of creating 3-dimensional objects through its library.
- Networking Interface – The networking interface provides a protocol for what information to send or receive from user to server. The server, in the case of Counter-Strike: Source [197],

orders events received from the many players connecting and interacting with each other on a map. The event ordering of the server is final and clients are updated on their effects. Using such logic ensures players with a faster connection speed do not gain an unfair advantage on slower players [198].

- Game Logic – The game logic defines the goals and rules of the game.

A traditional game such as Quake [199] facilitates voice and text communication between players. Quake uses the physics engine Havok [200] to ensure virtual representation of objects behave as they would in the physical world. Users are represented *in-world* by an *avatar*, often chosen from a fixed list of characters. User customisation of their *avatar* is typically low. Game engines are also limiting in their ability to allow users to fully shape and create their environment. Many multi-user games only allow the use of pre-constructed elements such as weapons or vehicles but will not allow the user to create their own and then share that with other users.

World of Warcraft [201], Counter-Strike [202] and Quake [199] do not let users define behaviours of objects through the use of scripts. Some rudimentary scripting can be performed such as binding different keys to perform a set action *in-world*. An example of scriptable actions is moving forward, backwards, *inventory* selection and jump. The ability to script objects is unavailable and as such they behave in accordance with the physics engine and the level designers' original intention. Many game developers release editing tools that allow a user to build a map of their own and share this with other users [203-206]. Such tools require extensive familiarisation and have many difficulties. Changes made to the map can only be achieved in the editor. Once made, the map must be recompiled and again distributed to other users who need to overwrite the existing map locally. In this way, game engines do not readily support the standard MUVEs features of a collaborative environment, construction and manipulation persistence. Table 10 provides a summary of the views discussed in this section.

	MMORPG	**Online First Person Shooter**	**MUVE**
***In-world* user-designed *avatar*s**	×	×	✓
Voice and text messaging	✓	✓	✓
Ownership of hardware for world	×	✓	× - Second Life ✓ - Open Simulator
Creation of elements *in-world*	×	×	✓
Scripting *in-world* elements	×	×	✓
Example titles in genre	World of Warcraft Aion [207]	Counter-Strike: Source	Second Life Open Simulator

Table 10: Comparison of typical MMORPGs, MUVEs and online real time first person shooters

Table 10 highlights some aspects which are significant for learning. Firstly, the ability for voice communication can aid remote students to easily communicate with each other. The ability to send text messages between students enables the transfer of information and helpful instructions from student to student.

EVE Online [208] is a massively multiplayer game enabling all players who join to be a part of the same game world. Players compete in a science fiction style outer space setting where they can create custom spacecrafts and weaponry. This is in contrast to many other games which, whilst large in the number of players online, cannot facilitate all the users inhabiting the same world and instead they play in parallel servers. A smaller example is Counter Strike [202], where millions of users connect to many public and private servers where a maximum of 32 or 64 can play. The state of the player is not maintained in Counter Strike, no global record of deaths or achievements is saved, unlike EVE online where the player's achievements are saved. Many virtual achievements can be sold for significant sums of money in the real world [209]. There is an extensive economy used in EVE, which enables players to buy and sell merchandise as well as the ability to create *in-world* items.

World of Warcraft is a MMORPG (Massively Multiplayer Online Role Playing Game) and is the most popular currently in use, with more than 11.5 million subscribers [201]. World of Warcraft involves completing quests for experience points in a 3-dimensional world which interacts with every other player. Full character customisation and the ability for players to form alliances with each other expose the great ability of a MUVE for collaboration of like minded or goal driven groups. These massively multiplayer games require extensive networking research [210] to enable such advanced interactions between large groups of people.

Active Worlds [211] is a 3-dimensional virtual world that can be accessed via a user's web browser on various platforms. The user interface enables *in-world* interaction and construction of structures. Active Worlds lets users create entire 3-dimensional worlds that can be explored by other users. The initial release date was 1995 and was designed as an alternative to web browsing wherein users would create buildings to house and display information for users. Users are subdivided into Tourists, Citizens and World Owners. Tourists can enter for free but cannot save their usernames and have a limited choice of *avatar*s. Citizens pay a monthly subscription and may use bots, file transfer and voice chat. A World Owner is given their own environment on which they can build objects, manage who comes into their world and, upon extension, purchasing a galaxy or universe. The environment has been used by Quest Atlantis [212] to provide a learning environment for 9-12 year olds.

HiPiHi [213] is the first Chinese Multi User Virtual Environment founded by HiPiHi Co. Ltd in 2005 and with a beta release made available in 2007 [214]. The initial version of HiPiHi did not include any scripting abilities and no information on the capabilities or expected date for public access has been provided. The target audience is Chinese and a guiding principle for the lead developer has been to "hope it can embrace our own culture" [215]. All text chat is filtered for prohibited words and the information shared with the government which appears to undermine the founders hopes of "creat[ing]

a perfect society - a shared and fair world which will finally embrace the various cultures of the real world". Revenues are made by making land available to purchase and displaying *in-world* advertising based on search queries. Connection to HiPiHi servers is extremely slow and with consistent failures. There has been no scripting language made available to users and as such user creativity *in-world* is limited. While this may increase in popularity, use will be restricted to China only.

Open Simulator [216] is an open source effort, released under BSD license, to create a 3-dimensional virtual world. Multiple virtual worlds can be hosted by any person for free. Open Simulator is written in C# and is very open for plugins such as new physics engines. A useful feature is that the client-server protocol is managed through libsecondlife [217] which allows for the same client to be used for Open Simulator server as a Linden Second Life server. The open approach to virtual world construction and linkage provides a blank canvas for designing *in-world* applications. A problem with Open Simulator is that is in its early stages of development and this causes difficulty working in an ever-changing environment.

Open Simulator recognises that scripting is a very important aspect of virtual worlds, allowing richer and dynamic interactions *in-world* but there is not a firm implementation of scripting yet. Current support is for C#, LSL and OSL though not all of the commands and events have been ported from LSL into Open Simulator. A script runs in a sandbox environment, i.e. without access to external resources. The script's execution should also not destabilise the simulator, as such certain aspects must be regulated, e.g. CPU time, memory usage and communication. Pre-emptive multitasking, where a script is interrupted after a preset time to allow other scripts access to the CPU, is being pursued.

Croquet is an open source cross-platform project to create and deploy multi-user virtual environments [218]. The virtual environments can be created by developers using the Croquet Software Developer's Kit (SDK) [219], the virtual world is then shared with its users through web deployment. Croquet provides a very powerful platform enabling software developers to customise every aspect of the virtual world, though there are several barriers to entry. One such barrier is pre-requisite experience with Squeak [220], an object-oriented programming language based on Smalltalk-80 and programming in OpenGL [196]. The programmer must consider the scaling of the application and methods to cope with such a problem. Due to these barriers, there has not been the same widespread adoption as has been seen in Open Simulator.

4.11.2 Linden Scripting Language

Linden Scripting Language (LSL) [221] is used to create scripts which control objects in Second Life and also to mediate data from the external Internet to inside the virtual world. Scripts can be attached to objects and can respond to actions from objects and users. Actions include, but are not limited to, movement, texture and colour changes, creating other objects, displaying particles and providing rich interaction with users.

A feature of LSL is *states.* Each script has a *default* state, which is entered when saved or reset. Other states include *touch_start* which is triggered by the simulator when the object is touched by an *avatar*. Scripts are edited using a notepad-styled editor with syntax highlighted *in-world* and then compiled. Compilation errors are reported to the user who must correct them before the simulator will save the script. Scripts, like objects, can be restricted using the Second Life permissions settings, which can protect intellectual property from being copied or modified by other users [222]. Once the script has been compiled into byte code it is uploaded to the simulator.

Memory Management

A compiled script with its stack and heap is referred to as an image and is allocated a total of 16 kilobytes in memory. Due to this memory limit each script must avoid large data structures and memory intensive functions. The image is executed inside a virtual machine ensuring that there is no opportunity to access other scripts or other information held in the simulator. The execution of multiple images is achieved using time-slicing, ensuring each script receives an equal portion of time.

A difficulty in Second Life coding is a memory-efficient requirement to avoid degrading the performance of the script itself or other scripts on the island simulator. The only way of testing a library call's memory footprint is to experiment in scripts using the llGetFreeMemory [223] command before and after execution. Currently this work is being undertaken by the community at large rather than Linden Labs, with sites such as LSL Wiki [224] providing a valuable resource for many LSL developers. The Second Life browser does not contain a debugger for use with LSL.

External Internet Communication

There are three technologies used to transmit information into and out of Second Life, Email [225], XML-RPC [226] and HTTP [227]. Any script in Second Life can send emails using the llEmail [228] API command in LSL. Below is the example usage of the Email API command as it would appear in a script within Second Life.

```
llEmail("tommy@cs.st-andrews.ac.uk", "Greetings","Hello, World!");
```

The specified recipient will receive the following text to their email account:

```
Subject: Greetings
From: c075132e-9aba-0101-b515-1759c4e2c35e@lsl.secondlife.com

Object-Name: EmailExampleObject
Region: Minerva Island (226304, 234752)
Local-Position: (85, 22, 23)

Hello, World!
```

Figure 23: Details of an email received form a script within Second Life

The restrictions imposed on the use of the llEmail [228] command are a 4 kilobyte maximum size and a delay of 20 seconds in the script. The email body contains information on the object that sent the email. The body of the sent email contains the object's Second Life key and position in Second Life as is shown in Figure 23.

Scripts can use the XML-RPC protocol to communicate with compatible services outside of Second Life using the *llOpenRemoteDataChannel* [229] API command. This command opens an XML-RPC channel and raises an event that is handled by the script. The script handles an event with the *remote_data_channel* state definition in the source. Data can be sent using the API command *llSendRemoteData [230]*.

When data is received on a set XML-RPC channel the simulator invokes the *remote_data* method. The simulator passes a range of parameters including an integer, the key for the channel it was received on, the value received and an identifier of the sender.

Restrictions are imposed on the use of the XML-RPC command with one active request authorised per script at a time. Multiple requests will overwrite each other if the script does not observe the delay and await the completion of the first XML-RPC request. Programmers have been advised not to use XML-RPC due to requests taking over a minute to complete and with most requests never being completed.

A script can perform a HTTP request and receive the resulting data by using the *llHTTPRequest* [231] API command. An object inside Second Life can initiate but not receive a HTTP connection. Bandwidth limitations are imposed, at a maximum of 25 requests per 20 seconds and a maximum payload of 2048 bytes. Data received through HTTP may not be saved as a texture into a user's *inventory* as this would evade Second Life's business model of charging for uploading and storage of textures. The use of a HTTP connection for this purpose is an active community discussion seeking resolution [232]. There is no simulator delay imposed on a script for calling this API command, unlike Email or XML-RPC.

4.11.3 LSL as a new approach to programming

There have been many approaches to creating programming languages. The first programming language was a series of punch cards that controlled the movement of a linen loom. The punch card approach evolved into a procedural language. This type of language provided a list of instructions that were executed by the machine. There were rudimentary ways of controlling the structure with the introduction of the GoTo command allowing sections of code to be reiterated or omitted. Object-oriented programming enabled extensive reuse of code and enhanced modelling of real world problems. For the Java languages, there are many libraries that fulfil common objectives such as enhancing time capabilities (Joda [233]), providing a collection of enhancements to the official Java SDKs (Apache Commons [234]) or database connection and manipulation management (Hibernate [235]). These libraries are extensively tested and developed by a community of enthusiasts, they substantially reduce the time required to write an application that utilises one of these technologies.

As programming languages have evolved, so has the method by which programs are written. Programs have moved from simple single-purpose machine manipulation into complex installations spanning multiple computer sites, controlling many components and high levels of interaction with hardware, software and users. The languages used provide more detailed type-checking and other possible hard

to detect bugs during compilation. The languages are clearly structured using standard notations for variables, instances and other components of that language.

As the scope and size of programs have grown exponentially, new methods to allow collaboration between programmers have been created. Development environments grow in capability and complexity with code generation, language inspection and low-level support for many different types of versioning software (CVS [236], Subversion (SVN) [237], git [238] etc). Programming languages such as C, Java and C++ can be developed in a range of editors and even compiled using a choice of compilers, with the end result executable across any other computer that supports the language standard.

Not only has there been development in the languages used and the method by which code is written, but also the methodology has undergone many radical shifts. There are an increasing number of approaches to software engineering such as Prototyping [239], Spiral, Waterfall and Rapid Application Development [240].

When LSL is viewed with reference to the detailed developments in programming languages and coding development aides, it is certainly lacking and may be viewed as a step in the wrong direction. LSL introduces a new programming paradigm. Scripts created for execution in LSL cannot exist in an ethereal state. Each script must be bound to a primitive in the virtual world. LSL allows the rapid creation of code through an intuitive interface and without the installation of any tools. Compilation is done when saving changes to the script and no script can be saved whilst there are errors existing inside of it.

A feature of LSL is the ability to control the actions and interactions of a 3-dimensional object within a virtual world quickly and easily. Many software applications such as DrQueue [241] allow the rendering of more intricate 3-dimensional objects. Enabling extensive interactions with users remains unexposed. LSL is an ideal platform for rapid development of a 3-dimensional representation of a project, allowing it to be interactive with users on a global scale.

Having developed numerous scripts and functions in LSL, it is clear that the main component missing from LSL is the lack of a standard way to call methods in other scripts. This prevents the widespread proliferation of substantial libraries. With the advances that Open Simulator is continuing to make, an example of which is the support for multiple scripting languages, it is unclear how Second Life can continue to compete.

4.12 *Summary*

This chapter has detailed the technologies and concepts which are used in the WiFiVL. Several simulators have been described with the focus on the network simulator used in the WiFiVL, ns-2. Some of the functionality of ns-2 has been discussed, specifically the wireless networking capabilities. Due to several user interfaces being described in this dissertation, a section on usability principles has

been provided. Important heuristics in the field of usability have been detailed and these principles will be referred to in future chapters.

A concept explored in this dissertation is teaching the IEEE 802.11 protocol. This chapter has detailed some aspects of this protocol. An influential technology in this dissertation is the use of multi user virtual environments. This chapter has given a definition and an introduction to the variety of virtual environments available.

5 WiFiVL Design and implementation

This section describes the WiFi Virtual Laboratory (WiFiVL) Mark I and II, objectives, development, implementation, deployment and evaluation. The goals of WiFiVL are to:

- provide background and context information on the IEEE 802.11 protocols
- provide a learning environment that allows the construction of WiFi scenarios with minimal usability barriers
- run created scenarios on a high quality simulator
- construct a system that enables future adaptability and reusability
- enable anytime access to the learning resource

The goals of WiFiVL specified above seek to enable an exploratory learning approach. Section 2.3.6 suggested task-oriented exploration as the best method for learning. WiFiVL seeks to enable this task-oriented exploration through the provision of a worksheet to provide attainable goals for the students who use the tool. The design of WiFiVL is such as to encourage a open learning environment where learners can formulate principles as suggested in section 2.3.6.

The WiFi Virtual Laboratory has been iteratively developed, utilising feedback from over 100 users and incorporating the heuristics of usability. WiFiVL Mark I utilises a HTML form to enable the user to define a scenario. The scenario is simulated through the reusable framework described in this section and animated to the user through the use of Flash.

WiFiVL Mark II utilises the same framework but contains an alternative user interface. A Rich Internet Application (RIA) is used to give a more intuitive interface to the user. Scenarios are constructed using point and click actions and the playback is viewed in the same container.

5.1 *WiFiVL Host Site*

The host site has been developed over several years with contributions from multiple authors. On the static resource host site, information on the wireless protocols is conveyed using interactive hypermedia. Examples of the animation are shown in Figure 24 and Figure 25. Figure 24 is the exploration of the hidden node problem through a Flash animation which shows the point of collision.

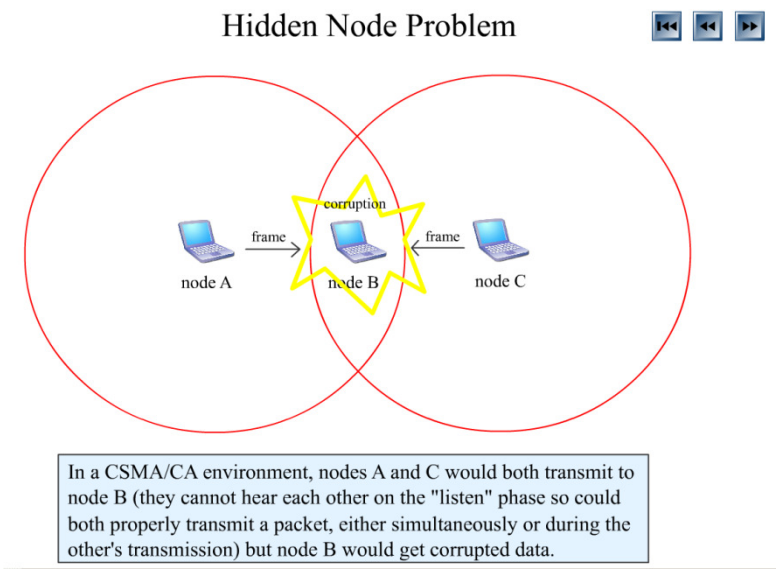


Figure 24: Screenshot showing a flash animation explaining the hidden node problem, available on the WiFiVL host site [242]

The host site allows the exploration of a wireless frame using Flash. A historical context of wireless networks is provided which is also supplied with interactive animations.

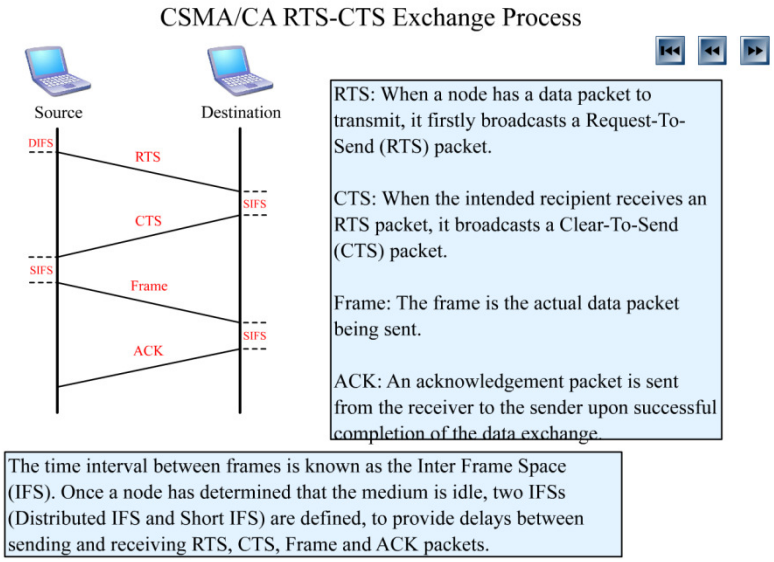


Figure 25: Screenshot of an interactive hypermedia element explaining the RTS/CTS exchange

Table 11 shows how WiFiVL and its resource host site form complementary resources for learning features of the 802.11 protocol.

		WiFiVL Resource Host Site	WiFiVL
Interface and Computational Resource	Simulator		➳
	Animations	➳	➳
	Replicability	➳	➳
User Interactions	Hypertext	➳	
	Video Playback Controls	➳	➳
	Parameter Input		➳
	Automatic Simulation Generation		➳

Table 11: Comparison of learning resources

5.2 *Components in WiFiVL*

The design of WiFiVL is in accordance with the Model View Controller (MVC) architecture (see section 4.9). The MVC structure, in its application to WiFiVL design, is shown in Figure 26. Each of the components in the diagram has a clearly defined role and provides a clear way to interact with the model.

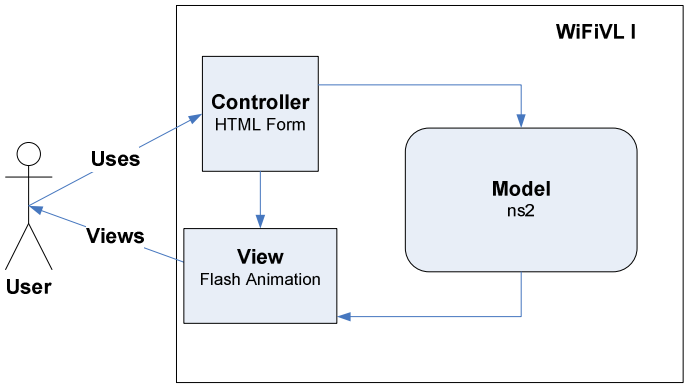


Figure 26: Model View Controller representation of WiFiVL

WiFiVL uses several components to translate the scenario specified by the user into a visualization of a simulation. Each step is detailed below and requires the use of multiple components which have been designed to be reusable. The sections below dissect component functionality, parameters and output.

A description of the approach used in the design and construction stages is also provided. A diagram of each component in the user request cycle and their interactions is shown in Figure 27.

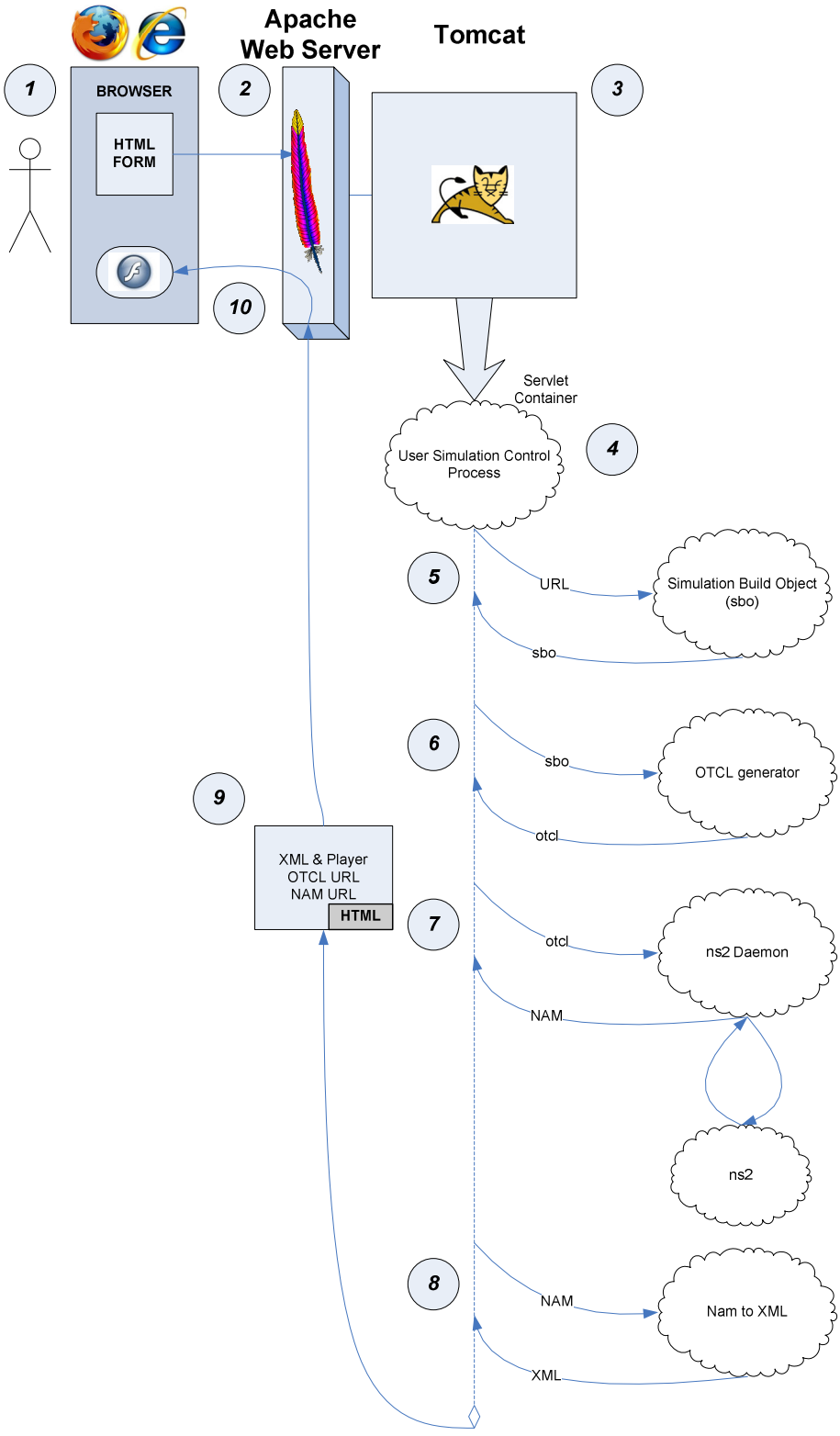


Figure 27: Components involved in WiFiVL

Step Number in Figure 27	Section Heading	Section Number
1-2	Scenario construction using HTML Form Elements	5.2.1
3-4	The WiFiVLServlet	5.2.2
5-6	jOBA – Java OTCL Build API	5.2.3
7	ns-2 Client and Daemon	5.2.4
8	NAM to XML Builder	5.2.5
9-10	Flash Animation, Graphical Representation of the Simulation Results	5.2.6 - 5.6

Table 12: A mapping of the numbered items in Figure 27 and the chapter heading and number that discusses that concept in detail

Figure 27 shows a high level view of the components in the WiFiVL and their interactions. In step 1 the user completes a HTML form which is used to describe a scenario for simulation. This description of a simulation is then encoded and sent to the Apache web server in step 2.

The web server invokes Tomcat through an HTTP-connector and the request is passed to the WiFiVLServlet, shown here in steps 3 and 4. This Servlet is used to control the input and output of the components.

In step 5 a Java object is created from the user's form and represents a model of the scenario. The object from step 5 is then converted to OTcl [22] at step 6. The output of the OTcl generator is passed to the ns-2 daemon in step 7. This daemon invokes ns-2 which simulates a given scenario. The output from the simulator is a time-ordered trace of every event that occurred during the simulation. The file is of type NAM due to the Network Animator application which uses the trace files. The NAM trace file is returned via the ns-2 daemon and is then used in the next step.

Step 8 invokes a component that converts NAM into an XML document. This file is then returned to the browser in step 9. The XML file generated from step 8 is animated using a Flash file in step 10. Each of the above steps is discussed in detail below in the corresponding section references as shown in Table 12.

5.2.1 Scenario construction using HTML Form Elements

The language used for storage and retrieval of user configuration of a scenario in a HTML form is JavaScript. A screenshot of the form interface is shown in Figure 28.

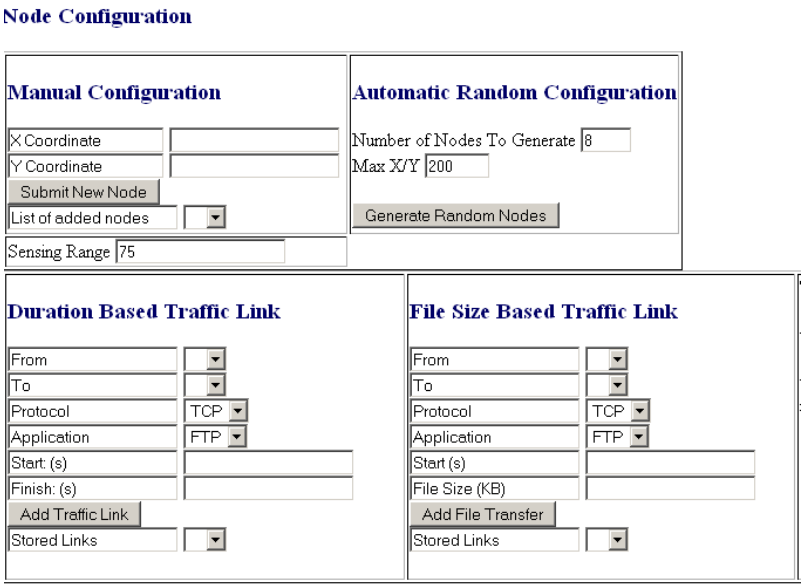


Figure 28: Screenshot of the WiFiVL way of specifying node locations, sensing range and traffic configurations

A user sets the location of a node by specifying the X and Y coordinates. Alternatively, a random distribution can be generated inside a specified maximum topology. Figure 28 is an example screenshot where the topography dimension is 200 metres with 8 nodes distributed randomly. As each new node is added, a representation of that node is stored using JavaScript. Each node has its own class representation as shown in Figure 29. JavaScript performs object orientation (see section 4.7) by using associative arrays. The user can review added nodes using the drop down menu labelled "List of Added Nodes".

```
function nodeObj(id, x, y) {
    this.xcoord = x;
    this.ycoord = y;
    this.id = id;

    this.toGet = function() {
        return "&node=" + this.id + "," + this.xcoord + ","  + this.ycoord;
    }
}
```

Figure 29: JavaScript object code that describes a Node

Once a user has defined two or more node positions, a traffic link can be constructed. The traffic links can be used to transfer data for a set period of time or to transfer a certain size of file. The applications and transport protocols can be specified by the user, e.g. FTP/TFTP over TCP/UDP file transfer. The JavaScript code used to model a traffic link is shown in Figure 30.

```
function trLink(src, dest, pro, appln, start, finish, itd) {
    this.source = src;
    this.dest = dest;
    this.protocol = pro;
    this.application = appln;
    this.startT = start;
    this.finishT = finish;
    this.id = itd;

    this.toGet = function() {
        return "&tt=" + this.source + "," + this.dest + "," +       this.startT + "," +
        this.finishT + "," + this.application +      "," + this.protocol;
    }
}
```

Figure 30: JavaScript object representation of a time oriented traffic link in the users HTML configuration page

The user input shown in Figure 31 exposes the advanced capabilities of ns-2. The advanced options are normally hidden but, in keeping with usability heuristics (see section 4.2), are available for power users of the system. The path loss, measured in decibels (dB), is the attenuation of the signal during transmission. Path loss has various causes including diffraction, reflection and absorption from the environment, such as walls or humidity.

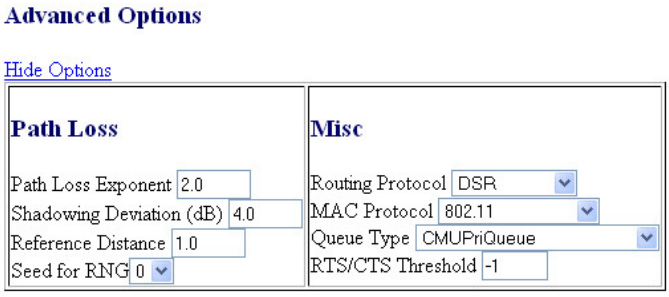


Figure 31: Screenshot of the HTML user interface; this section is contained in the Advanced area of the configuration page

The "Seed for RNG" option allows the user to choose a random number generator for the simulation. Setting this value to zero will ensure that when replaying a simulation it will produce the same output consistently. This replicability is educationally important as it allows the experiment to be reproduced with the same results repeatedly. Equally it is important to allow experimentation with random values to allow the student to test their hypothesis under slightly varied conditions. As such, the ability to introduce random events has been included.

Various routing protocols may be selected based on the users' requirements or interest. The different routing protocols are shown as part of Table 13.

The Request To Send (RTS) / Clear To Send (CTS) threshold allows for the addition and removal of the RTS/CTS mechanism in IEEE 802.11 [158]. This feature of ns-2 exposes the RTS/CTS mechanism which solves some of the problems associated with wireless networks (see section 4.10.5). Hence, the ability to enable and disable the RTS/CTS mechanism is pedagogically important and should be observable in both animation and statistics. Similarly, different MAC protocols have been

included for exploration by the user such as those in Table 13. The inclusion of other protocols enables students to experiment and construct a comparison with the IEEE 802.11 protocol.

When traffic communication between nodes has been configured, a request string is generated. The request string is constructed by iterating through all the stored objects that describe the scenario (nodes, traffic links and advanced settings). Each object has a *toGet* function that returns a string representation of its internal state. This representation is a shared protocol between the form and the WiFiVLServlet. The *toGet* function calls are concatenated together to provide a full description of the scenario and this is represented using a URL as shown in Figure 32. This request string is submitted to the WiFiVLServlet. The generated URL shown in Figure 32, in keeping the usability principle of supporting power users (section 4.2) can be manually edited by advanced users. This URL is the full description of a scenario and can be shared amongst students and teachers in a classroom or any other Internet user via social bookmarking, email, instant messenger or simple publication on a web page.

```
?&node=0,20,20&node=1,60,60&tt=0,1,0,1,FTP,TCP&rp=DSR&mp=11&qt=cpq&rct=1&sr=75&ple=2.0
&sdev=4.0&refd=1.0&seed=0
```

Figure 32: An example of a URL generated by the scenario construction from HTML form elements

A description of each of the parameter names and values is provided in Table 13 below.

Parameter Symbol	Parameter Meaning	Description and example values
node	Node	A node describes a simple wireless node involved in the network, it contains a Unique Identifier (UID), x-coordinate and y-coordinate: <UID,x,y>.
tt	Time Traffic Link	This traffic link will continue to transfer information at a set rate for the specified time. It has the format of source node UID, destination node UID, start time, finish time, application type and transport protocol. In the example given in Figure 32 the traffic link is between node 0 and node 1 which will start at time 0 (s) and finish at time 1 (s), the application is FTP and the transport protocol TCP.
ft	File Traffic Link	Type of link that will transfer a set amount of data between two hosts which will be initiated at a set time.
rp	Routing Protocol	The routing protocols available are DSR [243], AODV [244], Flooding and TORA [245].
sr	Sensing Range	Alters the value for the distance nodes can send and receive signals.
rct	Request Threshold	If the frame size is larger than this value then it will require a RTS/CTS prior to sending.
mp	Mac Protocol	The MAC protocols available include IEEE 802.11 [158], Aloha [246] and TDMA (Time Division Multiple Access).
seed	Seed	Provides the option of enabling random events.
qt	Queue Type	Various protocols can be used to deal with packet queuing at nodes. The different types are CMU Priority Queue, Queue/DropTail, Queue/DropTail/PriorityQueue .
refd	Reference Distance	This determines the value of each unit used for specifying node locations. So a 200x200 grid could be scaled by a factor of 10 by having a reference distance of 0.1.

Table 13: A description of the protocol used to describe a scenario as it is passed using a URL

5.2.2 The WiFiVLServlet

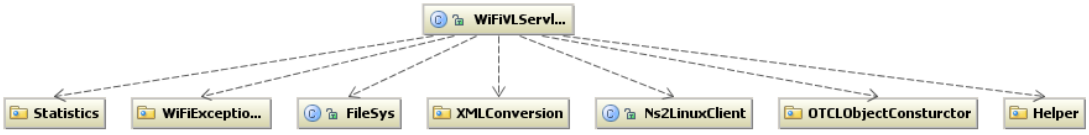


Figure 33: UML class diagram of WiFiVLServlet showing which packages are used

Servlet technology [247] is used to interpret the request string generated by user configuration of the form as described in section 5.2.1. The WiFiVLServlet controls the interactions with all the other components required by the WiFiVL. The WiFiVLServlet returns dynamically generated XHTML [248] pages of the resulting animation along with graphs showing the distribution of packet types.

The WiFiVLServlet takes the request string and mediates library calls to build an OTcl [22] file using the jOBA. This OTcl file is then made available to the ns-2 daemon using the ns-2 Client. The results from the simulation are transferred to the WiFiVLServlet. The Servlet then converts the results to an XML formatted document using the NamToXML Builder component. This XML document is used by the Flash player to display the network events to the user.

Each step in Figure 27 produces an output file. The jOBA produces an OTcl file, the ns-2 daemon produces a NAM file, the NamToXML components produce an XML file and the statistics components produces several graphs saved as a JPEG file [249]. Each of these files is saved by the WiFiVLServlet using a custom built FileSystem servlet. The files are stored in a temporary directory and referenced using a URL. The URL for each of these files is returned to the user, allowing them to be downloaded and used as part of the lab report, in alternative players or manually run in ns-2.

5.2.3 jOBA – Java OTCL Build API

OTcl [22] is an object oriented extension to Tcl [166] and is the language used to construct scripts that describe a scenario and initiate a simulation in ns-2. Using a programming hook an OTcl script can call C++ objects through the runtime interpreter. OTcl enables the construction of various simulation configurations, which, although allowed in the API, will cause a segmentation fault when run in ns-2. Such bugs are difficult to diagnose and resolve. No existing software was suitable for the aforementioned requirements and therefore jOBA was designed. The difference between this software and a scenario generator is that jOBA is an API. This API provides extensive documentation and library calls that allow an abstraction layer to the OTcl configuration file. The API has been extensively tested using JUnit [250]. The structure of the API is shown in Figure 34.

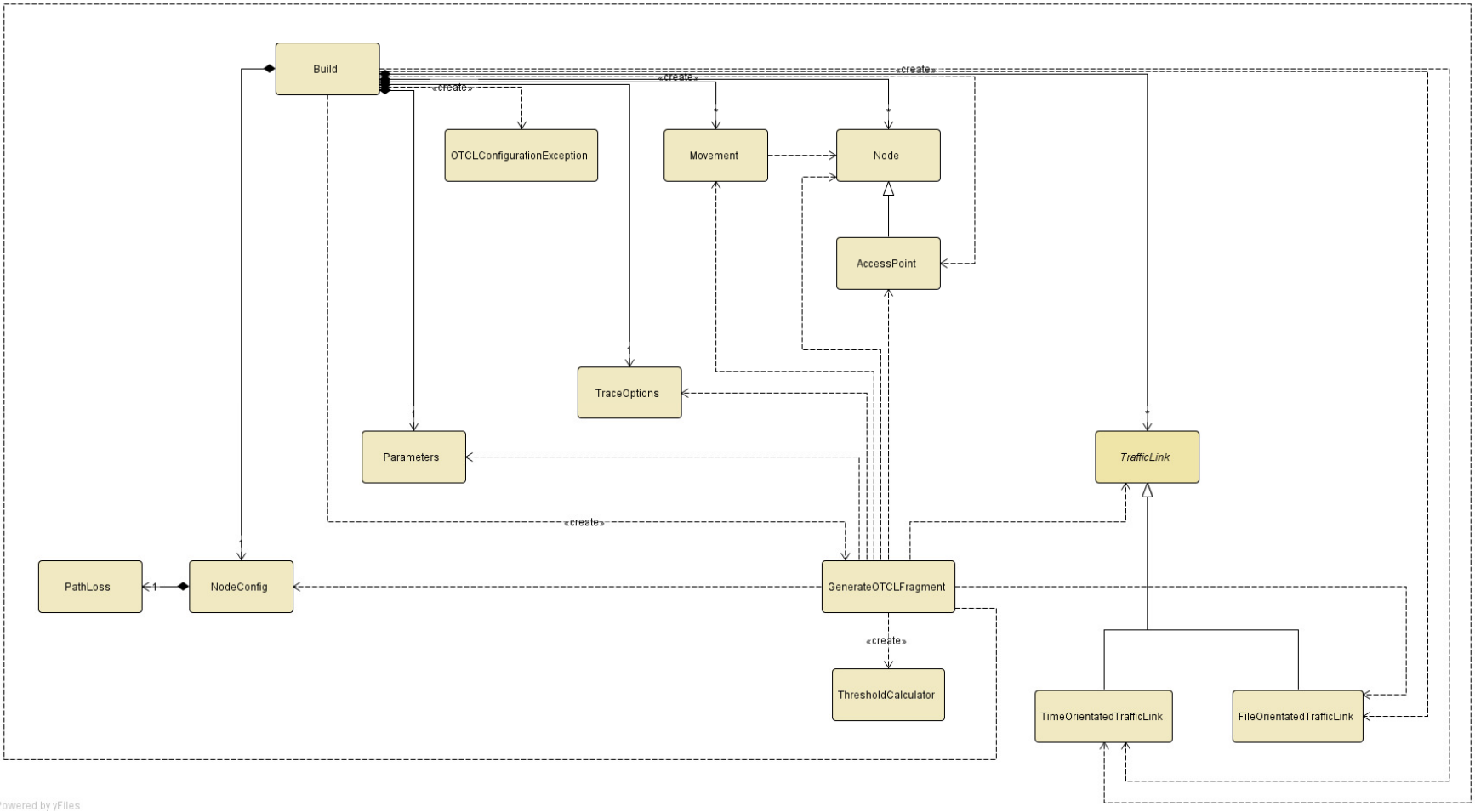


Figure 34: UML component diagram of the OTCL construction API

jOBA is a Java API that is instantiated by the WiFiVLServlet. The API dynamically creates OTcl files, which can be used in the ns-2 simulator. The jOBA is created by instantiating the Build class. To instantiate the class a range of parameters are required. The first parameter is an array of Node objects, which is a container for specifying the location and other features that are unique to the Node. The Node class has been extended to provide support for other ns-2 features such as a mobile node. Another type of node is an Access Point Node, which provides infrastructure support. Both of these nodes were added after the API was originally created. The extension was enabled by extending the class Node and adding in any special configuration required in the Node portion of the OTcl Generator. This new addition demonstrates the structured and flexible design of the API.

Settings, which apply to all nodes, are specified in the NodeConfig class file. This class allows the setting of a MAC type, routing protocol and sensing range. The RTS/CTS threshold variable is the minimum size of frame that will require a successful RTS/CTS exchange before sending. There are many different types of MAC protocols, routing protocols and propagation types to choose from. The choices are described to the programmer using enumerated types brought into the Java language with Java 5. An example of the MAC enumerated types is shown in Figure 35; this allows the programmer to specify a MAC type from a bounded set.

```
Public enum MacType {
        mac80211, macCsmaCa, macSat, macSarUnslottedAloha, macTdma
    }
```

Figure 35: MAC protocol choices shown as an enumerated type

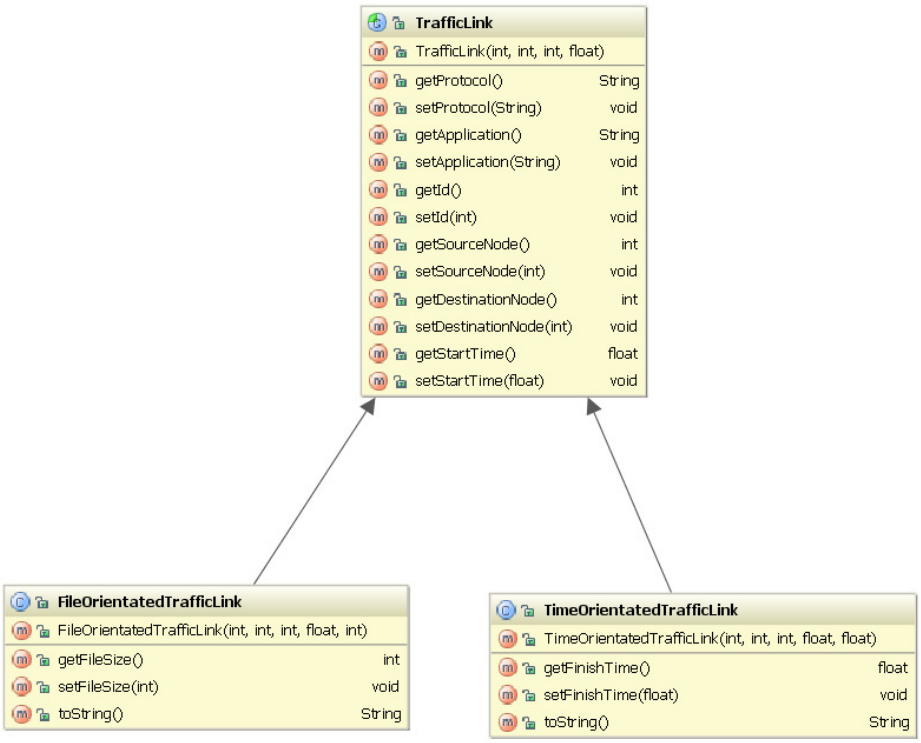


Figure 36: UML class diagram of a traffic link in jOBA and the inherited components, file and time transfer links

A parameter of the Build object is a Traffic Link which can be either a FileTrafficLink or TimeOrientedLink. Both types of links use protocols such as TCP [251] or UDP [252] for the transport level, and CBR (Constant Bit Rate)[3] or FTP [174] (File Transfer Protocol)[4] for application level protocols. The UML class diagram shown in Figure 36 represents the TrafficLink. This Java Bean [184] (see section 4.4) has getter and setter methods for the protocol, application, time for initiation, source and destination node UID. This class is extended to provide information for a file transfer and a time-oriented traffic link. This allows the higher-level construction of a Build scenario to ignore the type of traffic link requiring simply a list of traffic link objects. This also enables the API to be extended to provide other types of links between two nodes.

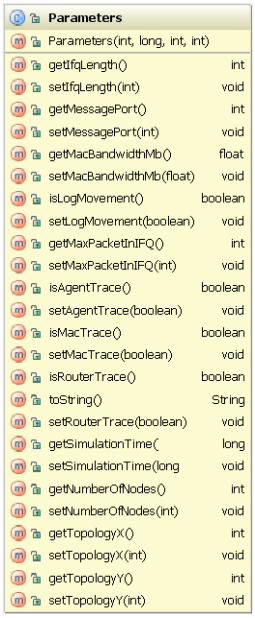


Figure 37: UML class diagram of the Parameters class in jOBA

The Parameters class, shown in Figure 37, contains methods that expose simulation settings such as the total simulation time, maximum packet length, topology dimensions and trace settings.

[3] Constant Bit Rate is a service that enables a continuous flow of traffic between two points at a specified bit rate.

[4] File Transfer Protocol is a common protocol in use for transferring files

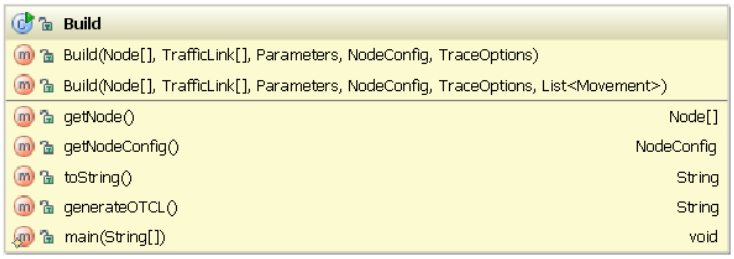


Figure 38: UML class diagram of the Build class in jOBA

The Build object is constructed using the classes shown in Figure 34. The Build class contains a method for returning the OTcl representation of a scenario as a script. The script can be saved to any output stream. The OTcl script is then generated using a method call in the Build object. The OTcl script is constructed through the concatenation of an object's internal state. These fragments are assembled and returned to the caller as a String. The OTcl data is written to a file and its location sent to the user.

The output is written to a file on the server, which is then made accessible through a URL. The use of URLs rather than passing around the information as internally stored strings is a design decision that allows the user to see each different file required for the simulation and can allow the student to take the OTcl file and alter it directly.

5.2.4 ns-2 Client and Daemon

ns-2 is a standalone package that is executed from a Command Line Interface (CLI) with the location of the OTcl file as a parameter. As part of the WiFiVL framework, ns-2 has been virtualised as a service, which produced a client and daemon. The ns-2 client allows multiple ns-2 servers to be used on a different physical location from the Tomcat server. The client is invoked through the WiFiVLServlet and the location of an OTcl [22] file is passed as a parameter. The OTcl file is referenced using a URL and is passed to the ns-2 daemon running on the same server as the ns-2 installation. The ns-2 client communicates with the ns-2 daemon through a TCP server socket. The client initiates the connection and sends a URL to the daemon; the URL is the location for the OTcl file. The ns-2 daemon then downloads the OTcl file, starts a thread to execute the script in ns-2 and then returns the contents of the NAM file. The NAM file is a record of every event in the scenario as generated by the simulator.

5.2.5 NAM to XML Builder

The NamToXml component converts the output of a ns-2 simulation, a NAM file, into an XML document. This is achieved with extensive reference to the ns-2 source code and manual [253]. The XML document provides meta-data allowing quick understanding of the network topology, parameters and packet types.

There are three main network configuration settings and network traffic events in the NAM file:

I. **Topology**: This set of data describes the size of grid ns-2 simulates. Only activity within these parameters are shown in the NAM file.

II. **Node Meta Data**: This set of data describes the transmission power, location, colour and shape of nodes.

III. **Packet Events**: A packet event involves new packets being sent, queued or de-queued in the network. Other events include the dropping of packets. Dropped packets may be due to collision, queue length restrictions or algorithms chosen.

The translation process involves reading the trace file line-by-line as shown in step 1 in Figure 39. Each line is one of the three types of events as described above. The line is then analysed and a Java Bean is selected that matches that lines instruction and returned as shown in step 2 of Figure 39. Step 3 is where the NAM line is transcribed into the Java Bean using its setter methods. Once all the required information has been obtained from the line and set in the Java Bean the toXML method is invoked. This will output the Java Beans state as a fragment of XML. This is added to the final XML file shown on the right hand side of Figure 39. This procedure is repeated for each line of the NAM file and once finished the XML file is returned.

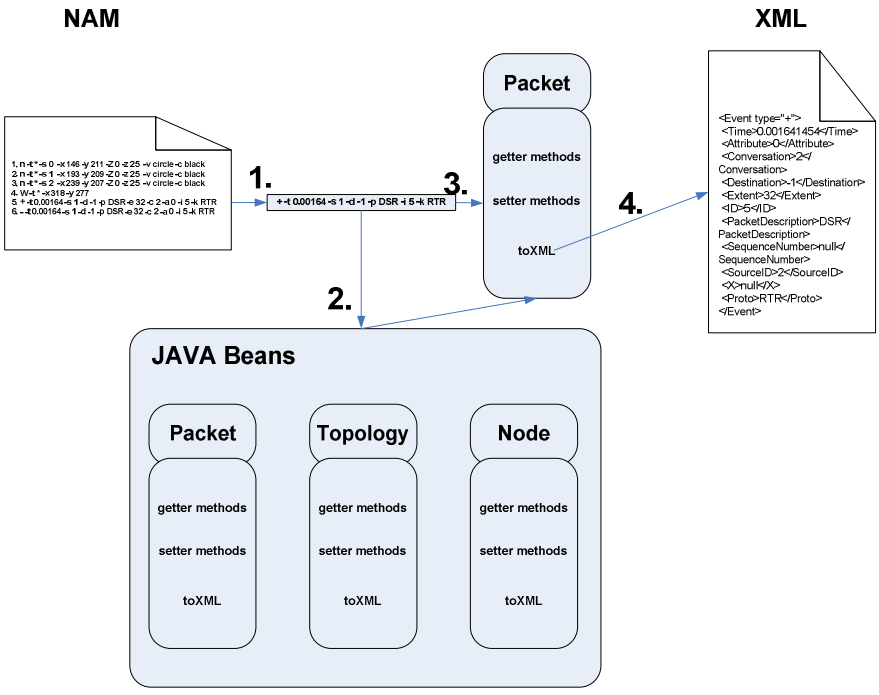


Figure 39: The translation of NAM To XML through the population of a Java Bean

The XML Schema used for the translation of the NAM file was designed to be human-readable and able to support a range of events generated in ns-2. Each Event that occurs in the NAM trace is translated into a suitable Event tag with an associated type. All of the Events in the XML file are child nodes of the root node *NamEventFile*.

Figure 41 is an example of the XML produced by the system from the NAM file in Figure 40. In this example the first Event node describes the configuration of the wireless network. The topology is set to 318 x 217 grid units. The second Event node describes the queuing of a packet at time 101.9 ms from node id 0 to the broadcast address and the packet is of type RTS (see section 4.10).

```
1.      n -t * -s 0  -x 146 -y 211 -Z 0 -z 25  -v circle -c black
2.      n -t * -s 1  -x 193 -y 209 -Z 0 -z 25  -v circle -c black
3.      n -t * -s 2  -x 239 -y 207 -Z 0 -z 25  -v circle -c black
4.      W -t * -x 318 -y 277
5.      + -t 0.001641454 -s 1 -d -1 -p DSR -e 32 -c 2 -a 0 -i 5 -k RTR
6.      - -t 0.001641454 -s 1 -d -1 -p DSR -e 32 -c 2 -a 0 -i 5 -k RTR
7.      h -t 0.001641454 -s 1 -d -1 -p DSR -e 32 -c 2 -a 0 -i 5 -k RTR
8.      + -t 0.002256454 -s 1 -d -1 -p DSR -e 84 -c 2 -a 0 -i 5 -k MAC
9.      - -t 0.002256454 -s 1 -d -1 -p DSR -e 84 -c 2 -a 0 -i 5 -k MAC
10.     h -t 0.002256454 -s 1 -d -1 -p DSR -e 84 -c 2 -a 0 -i 5 -k MAC
11.     r -t 0.002928608 -s 0 -d -1 -p DSR -e 32 -c 2 -a 0 -i 5 -k MAC
```

Figure 40: Example numbered output from the start of the NAM file

```
<?xml version="1.0"?>
       <NamEventFile>
              <Event type = "mediaConfig">
                            <tx>140</tx>
             </Event>
              <Event type = "nodeconfig">
                            <id>0</id>
                            <x>146</x>
                            <y>211</y>
                            <Colour>black</Colour>
                            <OColour>null</OColour>
                            <Shape>circle</Shape>
                            <SourceAddress>null</SourceAddress>
                            <State>null</State>
                            <Time>*</Time>
              </Event>
              <Event type = "nodeconfig">
                            <id>1</id>
                            <x>193</x>
                            <y>209</y>
                            <Colour>black</Colour>
                            <Colour>null</OColour>
                            <Shape>circle</Shape>
                            <SourceAddress>null</SourceAddress>
                            <State>null</State>
                            <Time>*</Time>
               </Event>
             <Event type = "wireless">
                            <XCoord>318</XCoord>
                            <YCoord>277</YCoord>
                            <Time>*</Time>
              </Event>
            <Event type = "+">
                      <Time>0.001641454</Time>
                      <Attribute>0</Attribute>
                      <Conversation>2</Conversation>
                      <Destination>-1</Destination>
                      <Extent>32</Extent>
                      <ID>5</ID>
                      <PacketDescription>DSR</PacketDescription>
                      <SequenceNumber>null</SequenceNumber>
                      <SourceID>2</SourceID>
                      <X>null</X>
                      <Proto>RTR</Proto>
           </Event>
           <Event type = "-">
                      <Time>0.001641454</Time>
                      <Attribute>0</Attribute>
                      <Conversation>2</Conversation>
                      <Destination>-1</Destination>
                      <Extent>32</Extent>
                      <ID>5</ID>
                      <PacketDescription>DSR</PacketDescription>
                      <SequenceNumber>null</SequenceNumber>
                      <SourceID>2</SourceID>
                      <Proto>RTR</Proto>
            </Event>
            <Event type = "h">
                      <Time>0.001641454</Time>
                      <Attribute>0</Attribute>
                      <Conversation>2</Conversation>
                      <Destination>-1</Destination>
                      <Extent>32</Extent>
                      <ID>5</ID>
                      <PacketDescription>DSR</PacketDescription>
                      <SequenceNumber>null</SequenceNumber>
                      <SourceID>2</SourceID>
                      <Proto>RTR</Proto>
            </Event>
       </NamEventFile>
```

Figure 41: Sample of XML text representing a wireless scenario converted from NAM type to XML. This is used by the WiFiVL for animating the result from simulating a given scenario

Figure 40 demonstrates that the conversion of the NAM file to XML in Figure 41 causes a large increase in the size of the space required for storage. Many of the performance penalties for increased space are offset by the GZip [254] compression algorithm which typically compresses by 95%.

5.2.6 Flash Animation

ActionScript [255] is a scripting language for controlling animations in a Flash file. Movies can be added or removed, and other graphics can be fully controlled from scripts. ActionScript also provides extensive support for XML processing. These features and the prevalence of the Flash player on modern browsers make it a suitable choice for rendering the resulting animation. An XML file containing a chronological packet trace of user-requested animation is accessed from the ActionScript code. One limitation of ActionScript 2.0 is the lack of threads and timing support. When displaying a series of events from the packet trace an accurate temporal system must be created. This was done by treating each new frame as a tick of a clock. The ticks' value can be altered to reflect different levels of time-granularity. Each new frame is rendered at a constant rate of 50 frames per second.

The second challenge is the creation of an execution scheduler in Flash. As each new frame is entered, its simulation time is calculated, e.g. if the new frame is 33 and the divisor 16000, then the simulation time it represents is between 0.002063 and 0.002125 (s). Any events that exist between those two time values must be animated to the user in the following frame. An event buffer of 20 events is pre-stored and when they are to be animated, they are pushed onto the execution buffer. The execution buffer is animated and cleared on each frame. The event buffer is refilled if it drops below a certain threshold, a diagram of which can be seen in Figure 42.

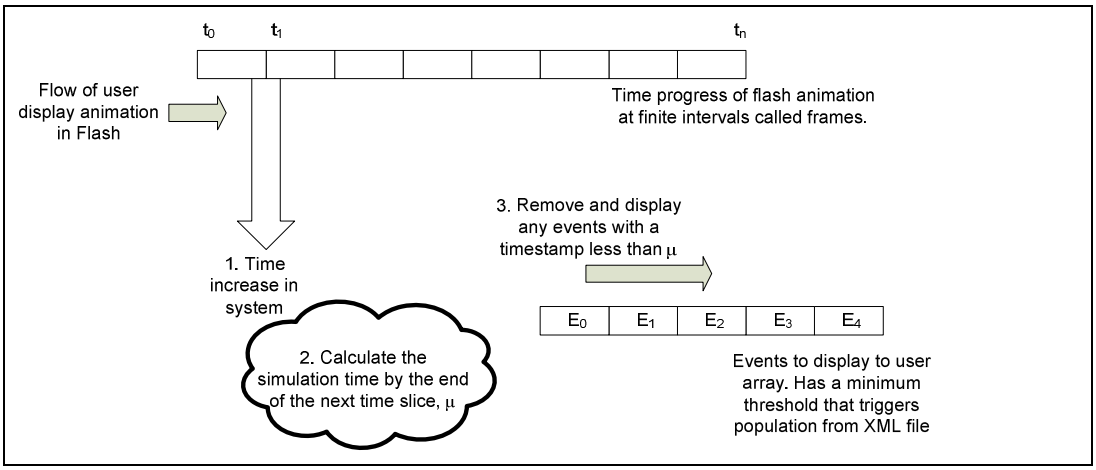


Figure 42: Execution stack implemented in ActionScript 2.0

The ActionScript code has been designed for extensibility, allowing further development in future work, such as different events, controls or different simulation types e.g. sensor networks and satellite communications. The screenshot in Figure 43 shows the layout of the WiFiVL player, which plays the XML file that is a trace of the simulation. In this example, two wireless nodes are transmitting an RTS packet simultaneously. The current time and speed stepper are in the top right hand corner. An event

history gives a textual representation that can be reviewed. Each type of signal is colour coded with the key in the top left hand corner. Basic playback controls of play and pause are located in the bottom left. Each packet that is sent into the network has its full details displayed in the box "Packet Info". Filters can also be applied to the animation so that only information at a certain protocol level is shown to the user in accordance with the usability principles outlined in section 4.2.

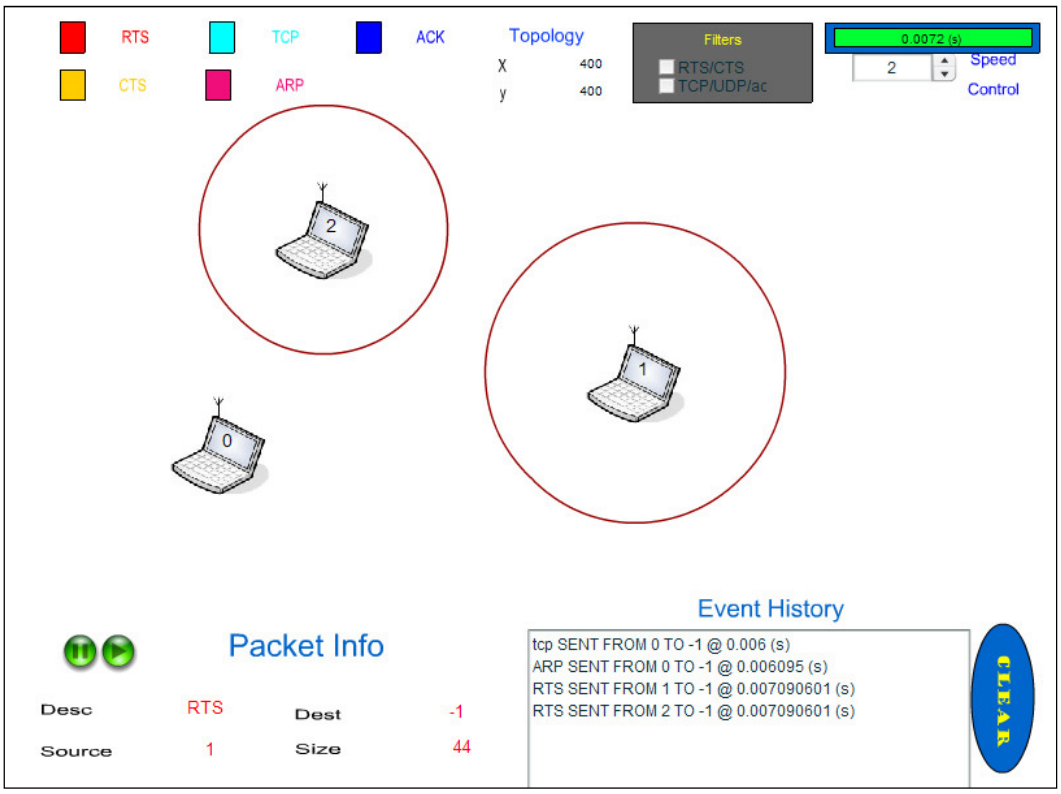


Figure 43: Screen capture of original WiFiVL showing the results from a simulation of IEEE 802.11

5.2.1 Education using Web 2.0

Current online social bookmarking applications allow easy sharing of URLs. These include Delicious [36], Digg [256] and Reddit [37]. These sites, and many others, allow users to create an account and save URLs that are of interest to them. Once saved, users can attach tags to be associated with the URLs, allowing for easy searching at a later point. URLs can also be shared between interested parties, users can watch another user's new URL submissions and add tags or notes of their own to a URL. Because the only requirement is a URL, the WiFiVL I, II and WiFiSL can have student scenarios saved. A student or lecturer who constructs an interesting scenario can copy the URL into their social bookmarking account. Tags indicating the type of educational scenario can be appended and shared with other students or lecturers.

5.3 *Evaluation of WiFiVL*

WiFiVL has been evaluated with students in the 3rd year module (CS3102) delivered as part of the BSc in Computer Science at the University of St Andrews.

The evaluation detailed here took place during a laboratory session for CS3102 in March 2007. The WiFiVL resource was available to students at any time. The students used Linux, CentOS with Firefox 1.5.0.9 on a university network and the worksheet provided in Appendix 1.3. The methods of data collection were: direct observation, interview, system log analysis, informal user feedback and formal feedback in the form of the System Usability Scale questionnaire. The students' performance on WiFi in their formal exam was compared to the previous year's cohort and also against other questions in the exam.

Some of the feedback indicated problems with the user interface. The flow of configuration, to start by defining node locations then traffic links between them, was not immediately apparent. A common theme was to try to break the system by adding thousands of nodes and trying to perform denial of service on the system by excessive resource utilisation. This was a recurring activity that continued throughout the laboratory session. However, this did not appear to have any impact on other users' simulation requests. The JavaScript that controlled the HTML form ensured that users could not make traffic connections between nodes that did not exist, but no limitations were placed on the number of nodes. This resource utilisation attack was prevented with a simple maximum node variable in both the JavaScript and in the server-side code. Users also attempted to enter coordinates with negative values. Such values were caught on the server side and similar detection was built into the client after the session. Despite frequent attacks the main web-server (Tomcat 4.1) and the ns-2 server, neither one crashed nor suffered a reduction in performance. This is noteworthy as the computer that hosted the ns-2 installation and a virtualised service was a moderately powered 1 GHz Pentium III, 20 GB hard drive, 512MB RAM and 2.6.x Linux kernel.

Some of the user feedback focused on the playback of the scenario, where the animated signals expand when a packet is sent onto the physical layer. It is expected that the packet will then expand to its transmission distance. Users raised the problem that the signal would not reach the other node on screen despite being placed within the transmission distance. This was a coding oversight and has been addressed. This interface discrepancy is an interesting point in demonstrating that users demanded accuracy and were engaged in discovering what the WiFiVL could show them.

Formal feedback was collected using the System Usability Scale (SUS) [257] which was used to evaluate the effectiveness of system interaction. The usability and perceived educational value of WiFiVL was compared to TCPView in [258] through SUS and Educational Value questionnaires. WiFiVL received lower scores for ease of use than the traditional static HTML page representation of a protocol (TCPView). This is not surprising as it is a more sophisticated and novel tool. Despite this problem the educational content was rated highly. In response to the question "I feel I have learned

something by using this system", an average score of 3.9 out of 5 was returned. Users also strongly agreed that "constructing and playing a simulation of the hidden terminal problem helped me understand it", returning an average score of 3.9. To summarise the results for the entire questionnaire, the upper quartile was 76.9, the median 67.5 and the lower quartile 50.0. This indicates that the learners found the system to be educationally beneficial. These results are summarised in Table 14.

Statistical Analysis	SUS Score
High	100
Low	10
Median	61.25
Mean Average	61.36
Upper Quartile	71.88
Lower Quartile	52.50
Spread	19.38

Table 14: Results of the System Usability Score for WiFiVL

The results of the SUS feedback are shown graphically in Figure 44. While it is clear that users thought that WiFiVL has educational value, there are some problems with the system usability. The SUS results were taken immediately after the 2-hour lab session and it is possible that the usability results may have improved if students had more time to become comfortable with the novel user interface.

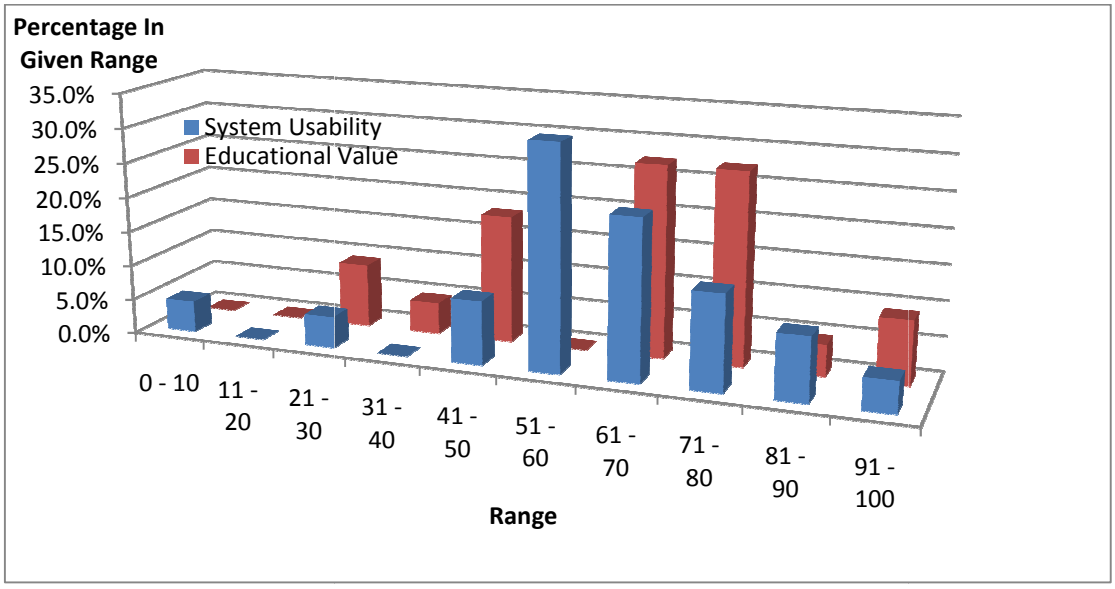


Figure 44: Educational value (red) and the system usability (blue)

Exam performance of a cohort of students from years 2003, 2005 and 2006 was compared to the cohort exposed to WiFiVL (there was no WiFi exam question in 2004). The results show that the number of students who performed at the highest level did not increase significantly but there was a noticeable shift in the students' performance in the lower bracket. Figure 44 shows the cumulative distributed frequency (CDF) of students against their exam performance. Figure 44 shows that the number of

students who performed poorly has decreased. This fulfils one of the principles outlined in this thesis: that it is beneficial to engage students through exploratory learning using simulation.

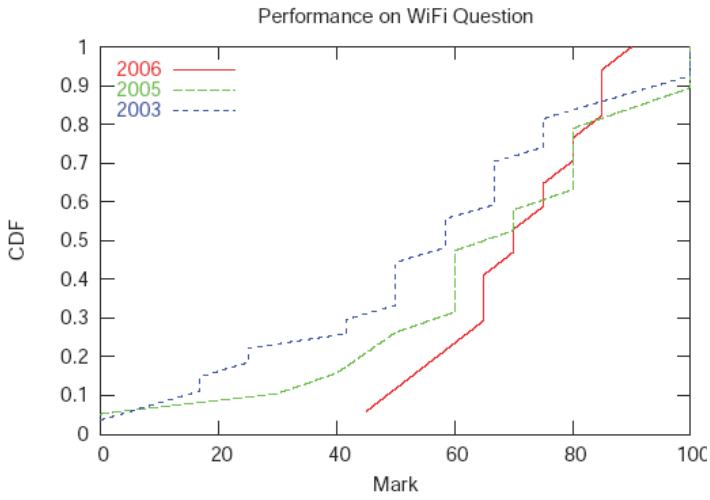


Figure 45: Graph showing the performance of students in an exam question on WiFi

Another area analysed in the evaluation was the performance of the class in the WiFi question compared with other questions in the paper. The results are shown in Figure 46.

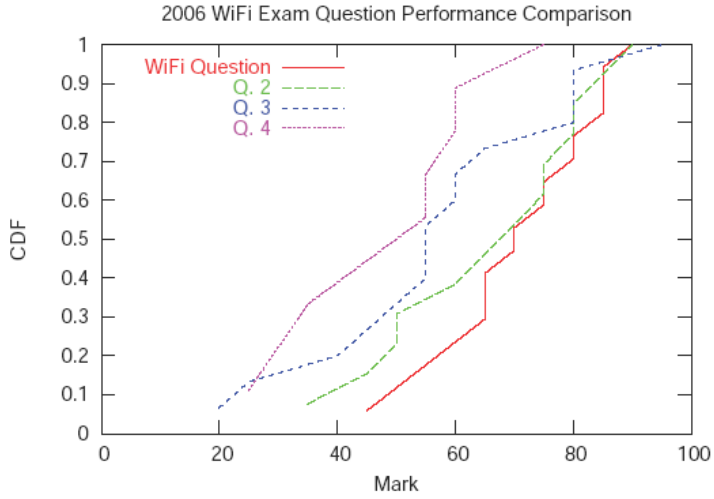


Figure 46: Graph showing the relationship between the scores for WiFi and other questions in a 2006 networking exam

In this graph, it can be seen that the lower bracket of students were performing much higher in the WiFi question than in other areas of the exam. The exam rubric stated that students should pick three of four questions provided. An observed effect was an increase in the numbers who elected to answer a WiFi question in the year that WiFiVL was used than in previous years. The results from this evaluation session are summarised in Table 15.

WiFiVL I	
Number of Participants	22
Evaluation Methodology	System Usability Scale, Perceived Educational Value, Exam Performance
System Usability Scale Average	61%
Perceived Educational Value Average	63%

Table 15 Summary of evaluation methodology for WiFiVL I

5.3.1 System Performance

A systems performance evaluation of WiFiVL was undertaken. Server logs recording each simulation request during a two-hour lab session of WiFiVL were analysed. On average a simulation was requested every 20 seconds. This pattern was later repeated using the same simulation requests submitted by the students. The time taken to generate the simulation and retrieve the output XML (Flash) file was measured. The results are shown below:

Rerun lab session			Rerun of single scenario	
Request Rate (ms)	Response Time (ms)		Request Rate (ms)	Response Time (ms)
21580	10		5	7.5
10790	13		2.5	6.5
7193	12		1	24
5395	37		0.5	41
2500	1000		0.25	41

Table 16: Results from re-running the scenarios requested by students during an evaluation session

The findings indicate that a single server such as a 1GHz Pentium III, 512MB RAM, 20 GB hard drive, producing a varied load of simulations at an average rate of one every 20 ms, can serve a class of approximately 80 students while maintaining a response time of one second. In the case of a simple scenario such as two nodes and one data transfer, the response time indicates that a much larger class could be supported without degradation.

5.4 *Extension of WiFiVL to WiFiVL II*

This section details the motivation for further development of the WiFiVL and a technical analysis of enhancements made. Further evaluation is provided which compares previous laboratory sessions where WiFiVL I was used to the new system detailed in this section.

What is demonstrated in the following sections is that WiFiVL II incorporates and extends WiFiVL I. Both systems work concurrently, accommodating the users' choice of preferred interface. The addition of multiple interfaces with the same underlying system and communication protocol demonstrates that the system has been designed with extensibility and flexibility. The extension of the existing Java OTcl Build API (jOBA) demonstrates that the API is extensible in supporting new features.

A roadmap for the proceeding sections is:

- Rationale for using a Rich Internet Application (RIA) framework and a comparison of existing frameworks detailed in section 4.8
- Design principles and methodologies adopted in the creation of WiFiVL II
- Extensions required in adapting the WiFiVL system to accommodate a new interface and capabilities
- Evaluation of WiFiVL II with students at the University of St Andrews

5.4.1 Motivation for developing WiFiVL II

Figure 44 shows that users rated the WiFiVL I as having educational value and this was reflected in the exam performance analysis detailed in Figure 45 and Figure 46. Figure 44 shows 4.5% of respondents' System Usability Scale (SUS) fall in the 0 - 10% range. The presence of low usability scale results and high educational value motivated development of an alternative interface, WiFiVL II.

Whilst some users responded with low SUS scores for the form interface, other users provided much higher SUS scores. This is evident from the spikes at both the low and high end scale in Figure 44. Therefore, in providing a more intuitive interface to facilitate more users, the original user interface should not be discarded. The choice of user interface should be left to the user and as such the HTML form and any other user interface built for WiFiVL II co-exist. The provision of multiple interfaces demonstrates that the system has been designed and implemented to be adaptable.

A Rich Internet Application (RIA) was considered the most appropriate means of providing an alternative interface because:

- A distributable binary built using C++ would limit cross-platform compatibility
- Many RIAs have extensive support for XML retrieval and processing
- XML is the format of the resulting animation for the WiFiVL

A detailed comparison of the RIA frameworks by Adobe Flex [259], Sun Microsystems (JavaFX) and Microsoft (SilverLight) is provided in section 4.8. The motivation for choosing Adobe FLEX was predominantly that it used a declarative XML (MXML) approach to constructing user interfaces, unlike Microsoft's Silverlight. There is also a widespread community using Adobe Flex unlike the more recently released JavaFX. Adobe released a stable, usable and intuitive integrated development environment (IDE) via an Eclipse [260] distribution dedicated to Flex. The special adaptation of the popular open source IDE Eclipse by Adobe provides developers with quick access to the tools and build SDKs required to develop and deploy a RIA using Flex.

Components used in WiFiVL (see section 5.2) have been extended and details provided in this section. The extensions to system capability detailed here are evidence that the code was adequately designed for reuse and extensibility. The WiFiVL II has been designed in accordance with the Model View Controller [190] architecture. The MVC approach is discussed in section 4.9. A diagram representing

how the WiFiVL II is structured in an MVC manner is shown in Figure 47. This diagram shows that there are multiple, interchangeable controls and views that represent the same model. The controller of views is also interchangeable (HTML form or RIA application), showing a strictly defined set of programming interfaces allowing such interoperability at the choice of the user.

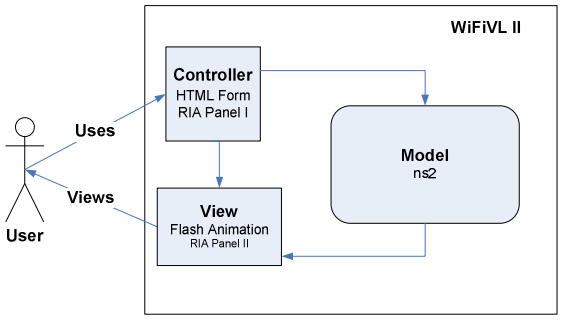


Figure 47: Model View Controller representation of the WiFiVL II architecture

The WiFiVL II user interface is described using the declarative interface language MXML[261]. This is a mark-up language that allows textual description of the user interface elements and its position on the screen for the user. In the example provided in Figure 48, it can be seen that two buttons are added called pointer and node. The resulting user interface elements can be viewed in the same figure. Buttons have actions associated with them which are invoked when a button is clicked. The checkbox labelled "Show Radius" is also added with associated actions that will be called when the user selects the box.

```
<mx:Button label="pointer" id="pointer"
        labelPlacement="left"click="deSelectAllButtons()"
        paddingLeft="10" paddingBottom="10"
        icon=" _pointerImg" y="72" width="70"
        height="40" horizontalCenter="-6"/>

<mx:Button  id="node" label="Node"   labelPlacement="left"
        click="nodeSelected(event);"
        paddingLeft="10" paddingBottom="10"
        icon="_nodeImg" y="140" width="70" height="40" />

<mx:CheckBox  x="35"  y="247"  label="Show  Radius"  width="110"
        change="toggleRadii()"selected="false" height="24" />

<mx:Button x="24.5" y="374"   label="Advanced Options"
        click="advancedOptionsButton(event)"  width="100"
        fontSize="8"/>

<mx:Label x="10" y="348" text="Sensing Range" width="89" />

<mx:TextInput  id="sensingRange"  x="107"  y="346"  width="41"
        text="140" />

<mx:Button x="24.5" y="475" label="Launch"
        click= "experimentalLaunch()" width="123.5" />
```

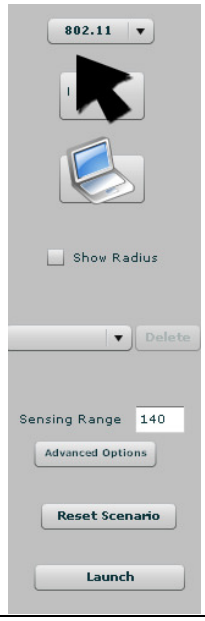


Figure 48: Excerpt of the MXML file that defines the user interface and the resulting user interface

The RIA for the WiFiVL II user interface has been designed to correspond to a set of heuristics providing consistency and familiarity for the user. An example of one such intuitive user interface is Microsoft's Visio [262]. Visio allows the easy addition of items to a design space. This is a familiar

concept to users and its adoption creates a lower barrier to system use. In the RIA for WiFiVL II, nodes are added by clicking on a node icon at the side of the screen. The button is shown in Figure 48. Once the node is selected, the user can click anywhere on the inner stage area to deploy the node. When a node has been selected, the cursor is changed to that of a node which indicates that the user interface is expecting to deploy a node to the main display area.

Traffic connections between nodes are instantiated by clicking on a deployed node, holding shift and clicking on a destination node. A dialog box appears, prompting the user for a transport protocol, start and finish time. Traffic links can be reviewed and deleted by selecting them from the drop down box which is populated by new connections created by the user.

Advanced options, such as selecting the routing protocol and setting the receiver threshold (further options and definitions are detailed in Table 13) are available, although they are not immediately visible to the user. When viewed, example values are provided, in keeping with the usability principle that the users should be given examples and defaults as to what values are expected (see section 4.2).

A "Show Radius" button lets the user view the transmission range of the nodes. This provides a quicker and more intuitive way to construct the hidden and exposed node scenarios (section 4.10.5).

Once the desired scenario has been constructed the "Launch Simulation" button can be clicked. The RIA uses ActionScript to construct a URL. As in the WiFiVL I user interface, a URL fully describes the scenario. The URL can be copied, sent to other users and then used to play back within either interface (Flash or RIA). This modularity and application of strict interface parameters provides the ability for multi-input and output interfaces.

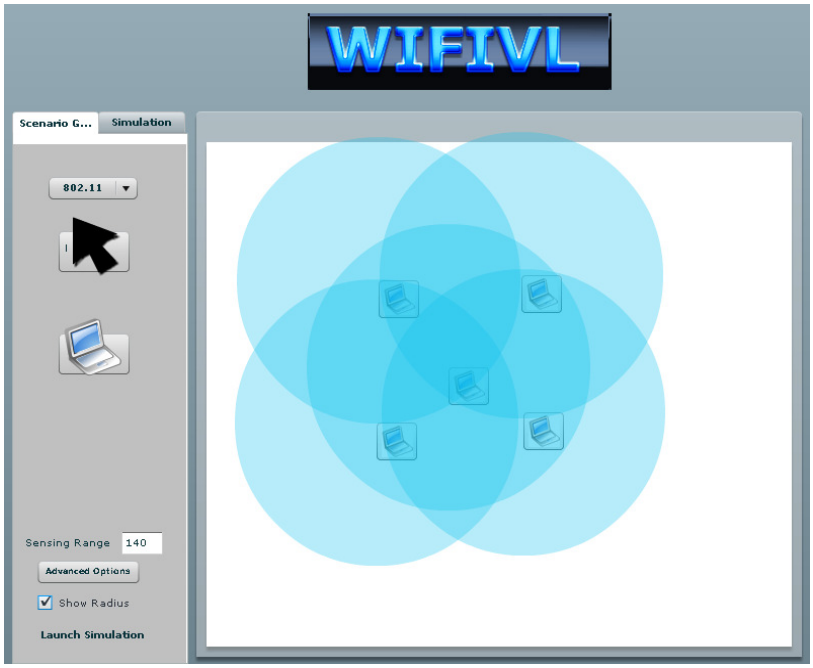


Figure 49: Screenshot of the new interface for constructing scenarios in WiFiVL

The WiFiVL Servlet was extended to provide multiple output types such as:

- nam – Outputs the NAM file from the simulation
- OTcl – Outputs the constructed OTcl file
- xml – Outputs the resulting XML which is the conversion of the NAM to XML
- stats – Outputs an array of generated graphs based on the NAM file

Each of the above extensions outputs the contents of that file to the requesting agent. This is further extended by appending '*url*' to the end which will cause the URL of the file to be displayed. This URL can be used in lab reports as a reference to the coursework that the student has undertaken. These output types enable more control of the output and allow integration with other systems, providing a service-based approach to the simulation.

Figure 50: Screenshot of the revised interface for defining a traffic link between two nodes. The pop-up configuration takes the focus of the screen, which gives a faded background appearance

The URL for the XML file that describes the whole simulation is passed into an ActionScript class. This class downloads the XML and preloads a set amount of events into a First in First out (FIFO) queue. This design approach is the same as was used in the previous animator described in section 5.2.6 Flash Animation and shown in Figure 42.

5.5 *Extension to jOBA*

New functionality has been added to the Java OTcl Build API (jOBA). The new functionality provides mobility and the inclusion of Access Points. The addition of these features allows for the construction

of mobile scenarios with access point association and disassociation. This functionality exposes some of the newer features of ns-2.33 and shows that jOBA can be extensible for new scenario types.

A node is defined as having an x and y coordinate only. This design holds for any node on a network regardless of whether it is an ad hoc network, access point or a node that can move. An Access Point is still a node on the networking requiring an x and y coordinate but also other new parameters. The object oriented nature of Java is used to extend the Node as shown in Figure 51.

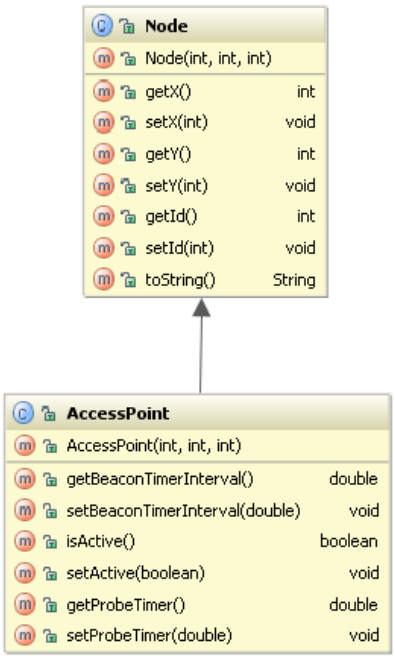


Figure 51: UML class diagram showing the enhancement of the predefined Node to allow for an Access Point

As shown in Figure 51 the Access Point class contains getter and setter values that define the presence or absence of an active beacon and its timer interval. This enhancement does not alter the main Build class which accepts a list of Nodes. This facilitates extensibility without having to rewrite multiple sections of code.

5.6 *Graphical Representation of the Simulation Results*

To enable quantitative comparison and discussion of contrasting simulations, a graphical representation of the network activity is required. Some examples of ns-2 charting [263] are complex and require a licensed installation of MatLab [264] software. A statistical analysis package for the WiFiVL II was therefore created; this is a modular package that accepts a NAM file as input. The output is several graphs created using JFreeChart [265]. These graphs are used to show what types of packets have been sent at multiple layers of the networking stack. For example a user wishing to investigate the benefits of RTS/CTS in IEEE 802.11 versus the Aloha MAC protocol can use the generated graphs to reference the differing levels of dropped packets due to collision.

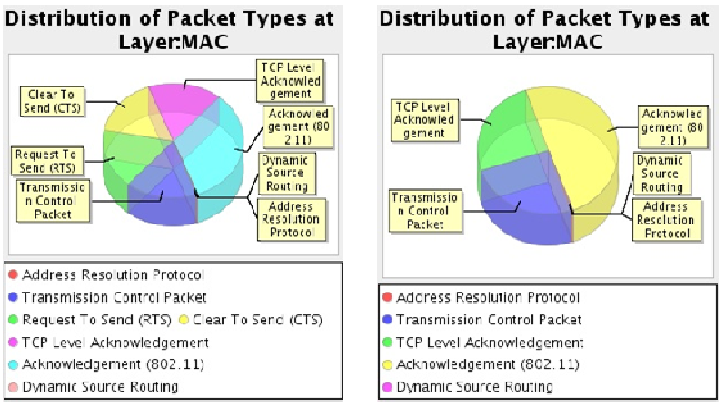


Figure 52: Chart showing the distribution of packet types. This is an example of the chart that is generated by a user-configured scenario in WiFiVL and shows how students can alter variables in the simulation to derive different statistical results

An example output is shown in Figure 52 and sample usage by students in 5.7.3.

5.7 *Supervised Evaluation Sessions for WiFiVL II*

There have been several laboratory sessions for both undergraduate and postgraduate students. This section focuses on selected sessions where the students provided evaluation data.

5.7.1 Supervised Laboratory Session of WiFiVL II

WiFiVL II was evaluated with students in a 3rd year module (CS3102) delivered as part of the degree BSc in Computer Science at the University of St Andrews. The evaluation detailed here took place during a laboratory session for CS3102 in May 2008. The laboratory session focused on the IEEE 802.11 protocols and utilised the WiFiVL II for exploratory learning. The students were provided with a worksheet that was used in previous laboratory sessions involving WiFiVL I. The document in Appendix 1.4 was provided to students as a guide to the new user interface.

During the laboratory session, malicious attacks were consistently crafted by a large proportion of the class. The majority of attacks were a type of denial of service through large simulation requests. Each denial of service attack consisted of adding many nodes and frequent, large data transfers in an attempt to tie up the simulator resource. The resulting output from WiFiVL II including graph generation, XML result construction, streaming to the RIA, event set-up and playback in the RIA was consistently within usability guidelines for response time of systems [266]. Other attacks consisted of modifying the URL providing invalid parameters to the WiFiVLServlet.

Students were also interested in the various routing protocols that were exposed to them through WiFiVL II. Many students' experiments involved varying receiver strengths and different routing protocols. Whilst most of the routing protocols were unfamiliar to them, research was performed using online resources such as Google, Wikipedia and the WiFiVL I host site.

The students easily constructed scenarios with increasing complexity and showed great interest in the protocols and software. Such complexity in simulations had not been seen in WiFiVL I where the only interface available was the HTML form.

The WiFiVL II question only occupies a quarter of the worksheet but became the focal point of the laboratory session. The students were keen to experiment with the software with most staying beyond the scheduled one hour session and continuing to provide informal feedback and creating new scenarios.

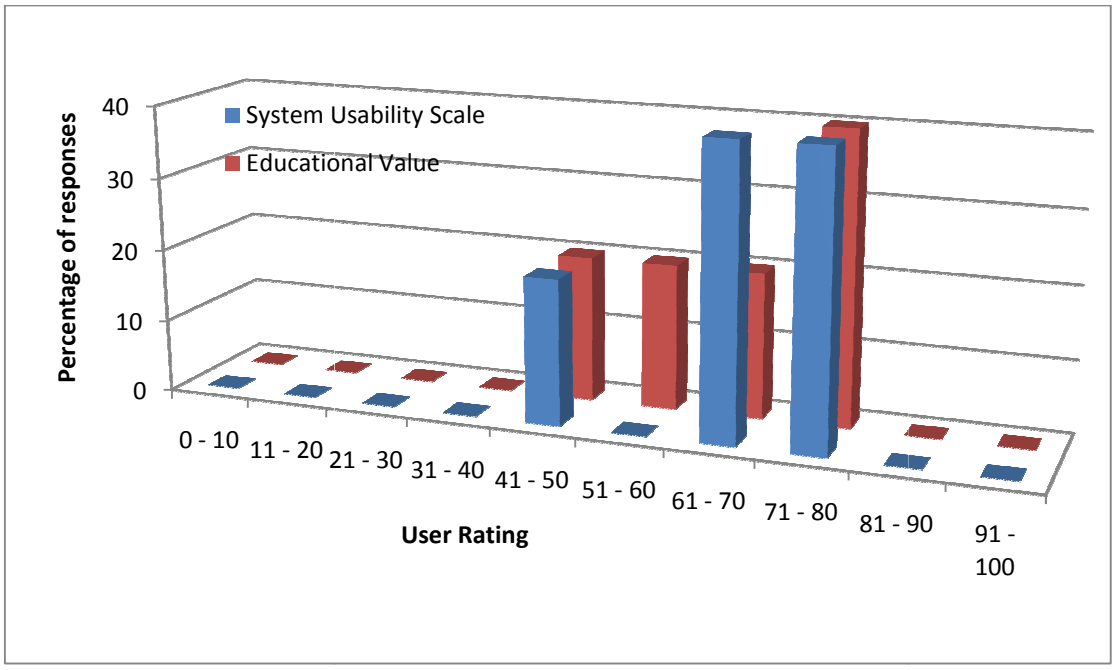


Figure 53: SUS and educational value in a laboratory session using the WiFiVL II. SUS average score was 61% and average educational value was 70%

Figure 53 shows a positive reaction to the use of WiFiVL II with a third of the responses giving a high educational value and usability rating of 71-80%.

5.7.2 Supervised Laboratory Session of WiFiVL II

WiFiVL II was evaluated with students in postgraduate module (CS5109) delivered as part of the degree MSc in Advanced Computer Science at the University of St Andrews. The evaluation detailed here took place during a laboratory session for CS5109 in November 2008.

The students were provided with the worksheet shown in Appendix 1.8. There were limited attempts to stress test the system by the students. This practical ran for a week, which presented an opportunity to observe the system over an extended period. The number of requests is shown in Figure 54. The system remained stable, with no crashes or service interruptions, and provided no decrease in response time.

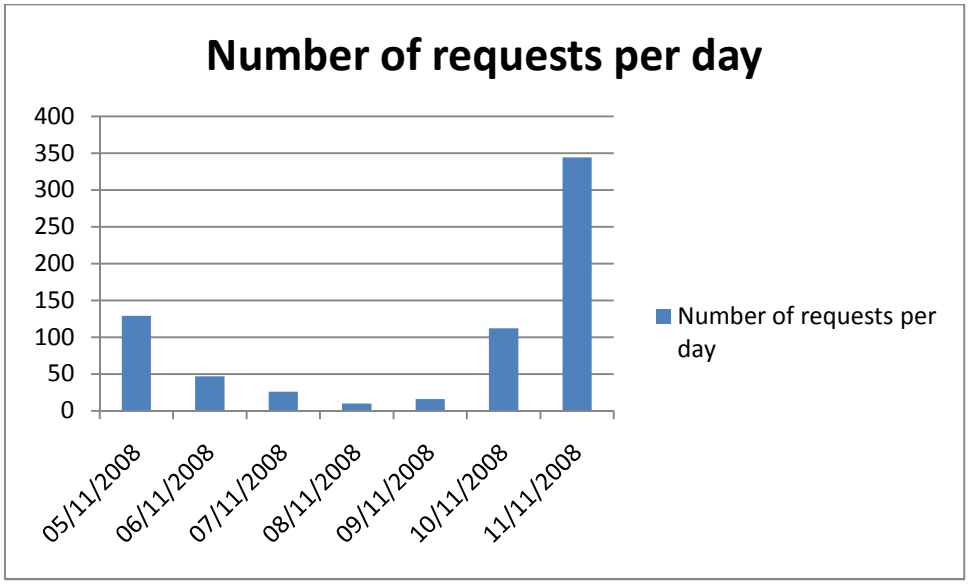
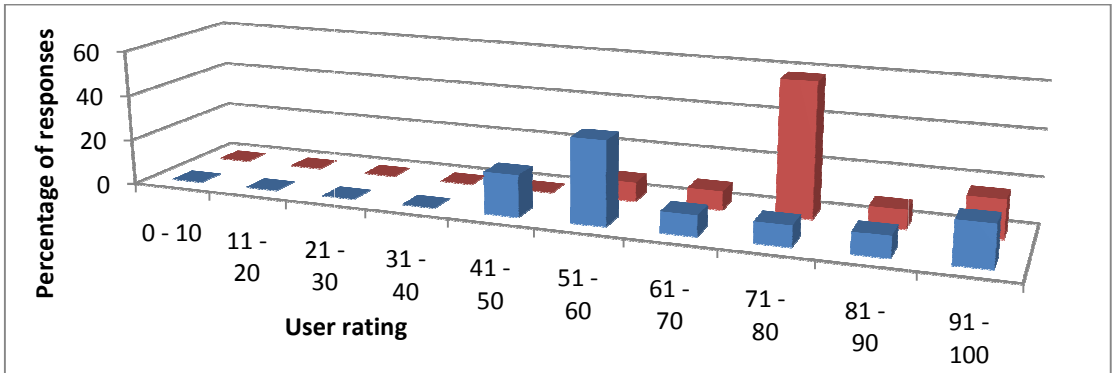


Figure 54: Usage of WiFiVL II over the week preceding a laboratory submission date of 2359 on 11/11/2008

A key success of the practical was the ability to use either a form or a point and click interface on the same system, allowing the student to choose the most suitable interface for their needs. The laboratory session also tested the ability of the statistics generation components of the system. The worksheet required an exploratory approach to learning in order to explain the changes generated in the graphs. One area for exploration was to investigate whether an increase or decrease in dropped packets occurred depending on the use of RTS/CTS signals.

The students were provided with evaluation sheets. The evaluation sheets took the form of a System Usability Scale (SUS) Likert scale for describing the interfaces available to them. Another evaluation sheet was provided to assess the perceived educational value of using the system. The results were strongly encouraging and are summarised in Figure 55. These results show an increase in the usability of the system and the maintenance of a high degree of educational consideration. The average rating for educational value is shown in Figure 56 and shows that WiFiVL II is seen as helpful in the education of the IEEE 802.11 protocols by the students. The question numbers can be found in Appendix 1.6.



	0 - 10	11 - 20	21 - 30	31 - 40	41 - 50	51 - 60	61 - 70	71 - 80	81 - 90	91 - 100
System Usability Scale	0.0	0.0	0.0	0.0	18.2	36.4	9.1	9.1	9.1	18.2
Educational Value	0.0	0.0	0.0	0.0	0.0	8.3	8.3	58.3	8.3	16.7

Figure 55: Distribution of SUS and educational value questionnaire. Average score for SUS was 66% and for educational value 79%

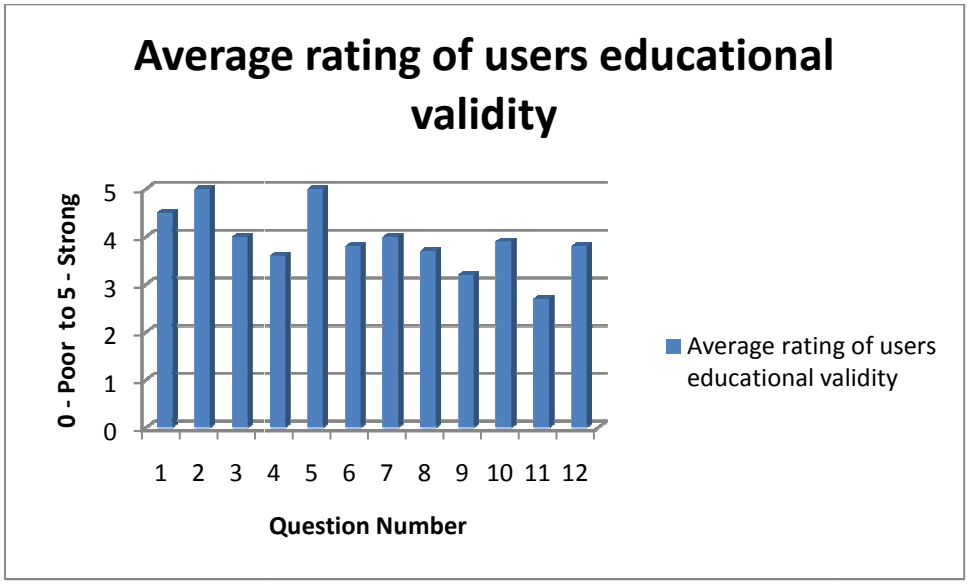


Figure 56: Average rating from the students on the educational aspect of WiFiVL II

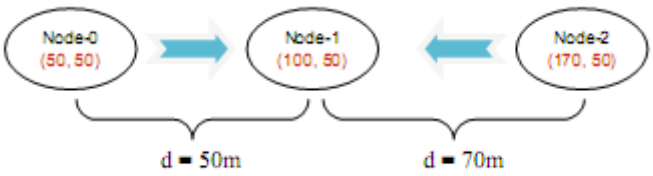
5.7.3 Example Student Report Using WiFiVL II

Figure 57 is an excerpt from a report submitted by a student asked to explore the hidden and exposed node problem as well as the effect of the RTS/CTS mechanism and its advantages and disadvantages.

3.b Experiments with WiFiVL.

The "hidden node" simulation:

The following environment was set up in WiFiVL for this scenario.



d = 50m d = 70m

Scenario parameters (in addition to node coordinates given above):

Sensing range = 75m

RTS/CTS Thresholds = 3000 (run 1) and 0 (run 2)

Size of file to transfer = 1024 KB

File transfer protocol = FTP / TCP

Node-0 begins a file transfer to Node-1 @ 0.005s

Node-2 begins a file transfer to Node-1 @ 0.015s

The following URL was used to generate this scenario:

http://wifi.cs.st-andrews.ac.uk:8080/WamNetMod/wifivl?&node=0,50,50&node=1,100,50&node=

2,170,50&ft=0,1,.005,1024,FTP,TCP&ft=2,1,.015,1024,FTP,TCP&rp=DSR&mp=

11&qt=cpq&rct=3000&sr=75&ple=2.0&sdev=4.0&refd=1.0&seed=0

As shown in the above diagram, Node-1 was placed 50m away from Node-0 and Node-1 was placed 70m away from Node-1 (along the same horizontal plane). As the sensing range was set 75m, Node-1 can receive signals from both Node-0 and Node-2, but Node-0 and Node-2 cannot sense each other's signals. The size of file to transfer was set at 1024KB to detect a significant number of dropped packets. Node-0 begins a file transfer to Node-1 at 0.005s and Node-2 begins a file transfer to Node-1 a little while later at 0.015s. The RTS/CTS Threshold was set at to a value of (3000) which effectively disables the RTS/CTS handshaking mechanism, as the largest possible 802.11 frame size is still smaller than this threshold value.

Observations:

When the above scenario was completed, there were 14 dropped packets reported by the simulator. Also, the file transfers from both nodes completed in about 0.0202 seconds.

However, when the scenario was re-[r]un with the RTS/CTS Threshold set to (0), the number of dropped packets reduced to 10. This was expected as a RTS/CTS Threshold of (0) means that every frame has to negotiate a RTS/CTS handshake before it's transmitted in the wireless medium. As such, the number of collisions is reduced, hence the reduction in dropped packets. Further, with full RTS/CTS handshaking turned on, the time taken for both file transfers to complete had increased to 0.263 seconds due to the overhead of the handshaking mechanism.

The following graphs depict the distribution of MAC frame types when the RTS/CTS. Threshold was set to 3000 as well as when it was set to 0. It can be seen that the ratio of dropped packets had reduced when the Threshold was set to 0 (i.e. RTS/CTS is enabled).

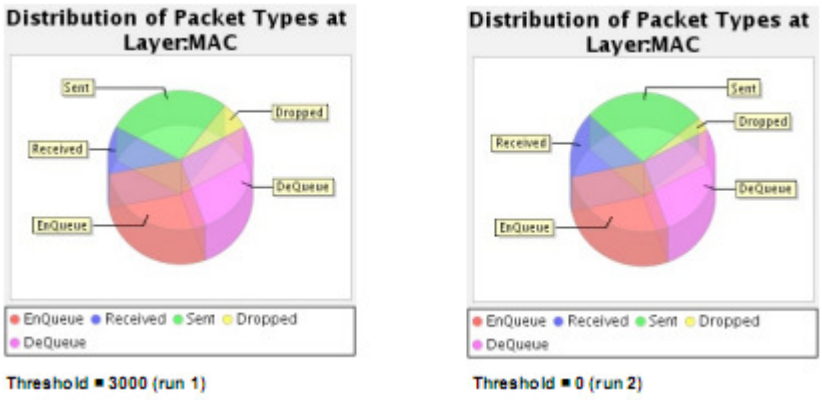
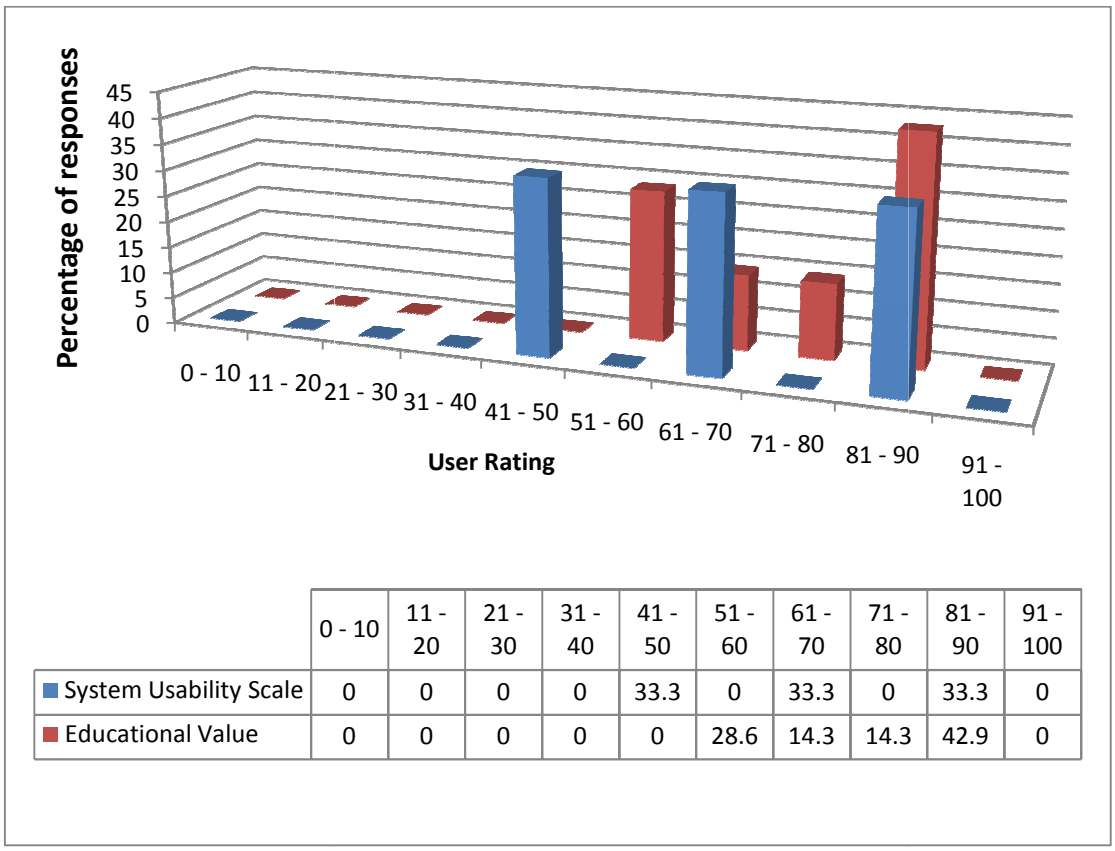


Figure 57: Example of a submitted student report

Figure 57 shows that the student has learned the basics and achieved an ability to replicate that basic information. They have proceeded to formulate a hypothesis and to test this out using the system. This hypothesis is then refined, proved and discussed, showing good knowledge of the IEEE 802.11 protocol.

5.7.4 Second Supervised Laboratory Session of WiFiVL II

WiFiVL has been evaluated with students in the 3rd year module (CS3102) delivered as part of the degree BSc in Computer Science at the University of St Andrews. The evaluation detailed here took place during a laboratory session for CS3102 in March 2009. The class size was considerably smaller than the evaluation session detailed in section 5.7.3 but the results in Figure 58 show no SUS scores lower than 41 and a large increase over the form only input.



	0 - 10	11 - 20	21 - 30	31 - 40	41 - 50	51 - 60	61 - 70	71 - 80	81 - 90	91 - 100
System Usability Scale	0	0	0	0	33.3	0	33.3	0	33.3	0
Educational Value	0	0	0	0	0	28.6	14.3	14.3	42.9	0

Figure 58: Results for the WiFiVL II System Usability Scale and Educational Value

These results are in keeping with other student responses to questionnaires when using the same version of WiFiVL and the worksheet.

5.8 *Conclusion*

This chapter described WiFiVL and its evolution including new output modes, mobility and access point functionality. New scenarios can now be created using the new RIA user interface provided.

Scenario construction is also possible using the older form based user interface. The bar chart in Figure 59 compares the results from student questionnaires for SUS and educational value of both the WiFiVL I and WiFiVL II systems. The solid dark red and blue results indicate the use of the form only in the first version of WiFiVL. The lighter red and blue are the results from the WiFiVL II where students can choose either a form or the RIA interface.

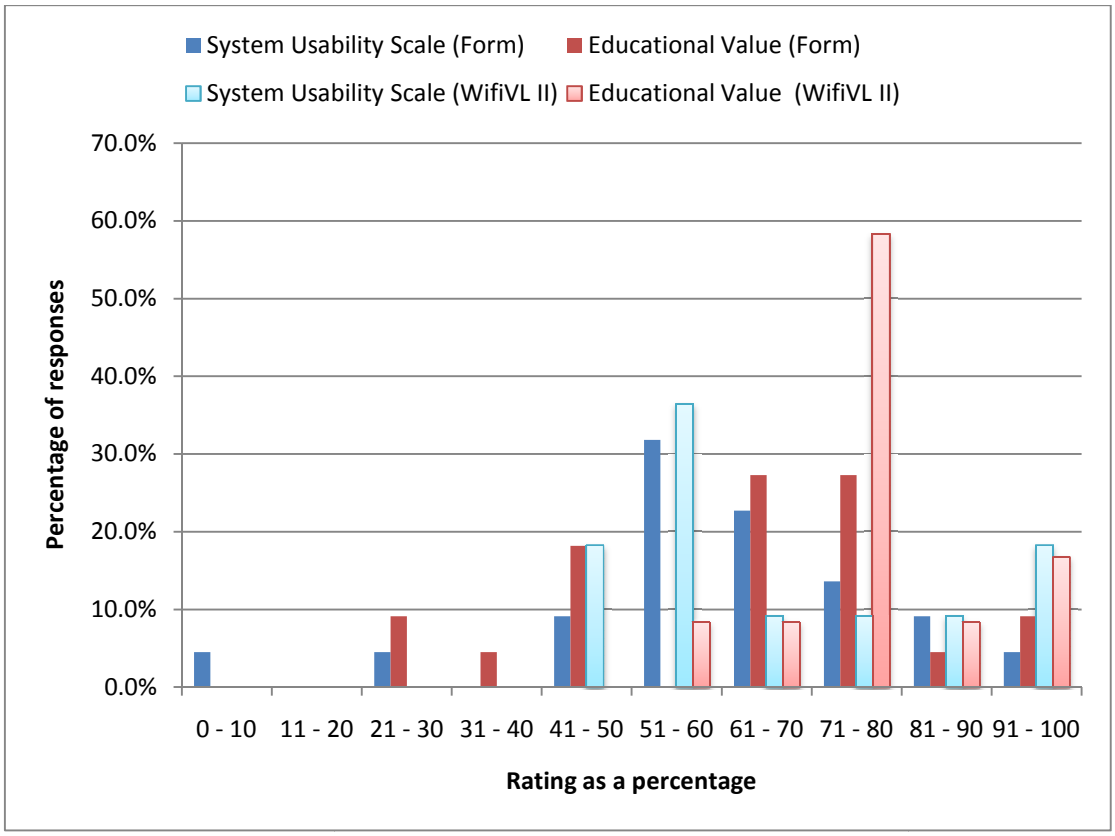


Figure 59: Bar Comparison of SUS and educational value questionnaire results when using just a form (dark blue and red) and the laboratory sessions where WiFiVL II was available (lighter blue and red)

The results from WiFiVL II show there are no SUS scores below the 41 - 50 range and an increase in the 91 - 100 section. Another noticeable increase is the users' answers to educational value. As the usability has increased and more features are added to WiFiVL, it becomes educationally more useful. Whilst it is impossible to say with absolute certainty that the WiFiVL II increases student understanding of IEEE 802.11, the results provided here certainly do not contradict such a statement.

A summary of the results of these evaluation session is provided in .

WiFiVL II	
Number of Participants	16
Evaluation Methodology	System Usability Scale, Perceived Educational Value
System Usability Scale Average	64%
Perceived Educational Value Average	76%

Table 17 Summary of evaluation methodology for WiFiVL II

These results show that the WiFiVL is worthy of further development and deployment in university networking courses. The system has been shown to cope with a class of 30 students. The results detailed here also show that the WiFiVL copes with a sustained long-term piece of coursework and has supported over 300 queries in one day without crashing or becoming congested.

6 WiFiSL

The use of emerging technologies is an apt way of engaging with students. The adaptation of networking protocols from the textbook to the web interface for WiFiVL I and WiFiVL II was motivated in part to engage with students. This section takes a broader view of the destination of user interests and interactions which may lead to widespread usage in future years. Computer Science students, who spend a large amount of spare time engaging online are often at the cutting edge of technology and are quick to adopt new and engaging mediums. Web 2.0 social applications such as Facebook [267] and Twitter [268] focus on user-generated content, communication and collaboration. Another emerging technology driven by user-generated content and communication is virtual worlds in the form of Multi User Virtual Environments (MUVE). An important heuristic in usability is that the interface should be enjoyable and this is very apt when using a MUVE. A virtual laboratory situated within a MUVE is a novel development of WiFiVL I and WiFiVL II and a method of continuing to engage and motivate students.

Recent advances of 3-dimensional virtual worlds offer new opportunities for researchers and developers. This chapter explores that potential. A virtual world is an immersive 3-dimensional environment where users can move around, explore, create and interact with objects *in-world*. This chapter details the design and development of a virtual wireless networking laboratory in a virtual world, WiFi Second Life (WiFiSL), and WiFi Open Simulator (WiFiOS). The concept of a virtual wireless networking laboratory has been explained and demonstrated in Chapter 5. WiFiSL and WiFiOS allow the construction of wireless scenarios in a virtual world. These scenarios are then simulated and the results animated back to the user.

One virtual island, Minerva, which is hosted in Second Life, is used firstly as a base for a WiFiVL. The second island is Aeolus, which is an Open Simulator installation and uses an entire island to provide a range of materials and activities for learning. These activities include traditional representations such as lecture slides, display boards showing historical growth of WiFi, the incorporation of streaming media and animated interactive displays. These resources are grouped together to provide an intuitive path of understanding that culminates in the WiFiVL where students can create their own scenarios and review a simulation in the virtual world.

The sections in this chapter will discuss:

- **Second Life and Open Simulator –** There are several implementations of a MUVE and a comparison of the main two environments is included. As a result of these discussions a virtual world is chosen in which to situate the WiFiSL and a detailed discussion of how objects are programmed is provided. A glossary of technical and *in-world* terminology is provided in appendix 1.1.

- **Design Considerations –** The actions and capabilities of a virtual laboratory will be described. This is a high level view of the types of interaction the user will perform in the virtual world to create and review a scenario.
- **Educational Scenarios –** Details of the range of scenarios that could be implemented and displayed in the virtual world will be set out. The list contains examples of educational scenarios currently implemented as a way of providing a general overview of the capabilities of using a virtual world for networking education.
- **Architecture –** A design for enabling users in a virtual world to create and review scenarios wholly *in-world* will be provided. Each component is introduced with a high-level description of their input, output and functionality.
- **Visualisation Considerations for Wireless Transmissions –** The use of a virtual world enables a new way of displaying wireless signals. Due to the novel nature of the laboratory there are few sources for reference. Several ways to display a wireless signal to a user will be described. Consideration is also provided for the capabilities of the chosen virtual world.
- **Traffic Construction Design and Implementation –** The user must construct a scenario in the virtual world by creating traffic links between networking components. This section describes a method for enabling an *avatar* to create a traffic link between two networking components in a virtual world. Partial source code is provided for the implementation of the traffic construction protocol.
- **Animation Controller –** Once a scenario has been described, it must be sent to the WiFi Constructor and Simulator (WiFiCS) for simulation. The result of the simulation is a discrete series of time-ordered events to be displayed. The animation controller displays signals in the virtual world. Each animation must have a source networking component and type of signal. The protocol designed to allow signals to be animated in a virtual world will be described in this section.
- **Evaluation and Methodology –** An overview of the approaches to evaluating a system with users will be provided. There are several protocols described and reflections on appropriate evaluation conditions and subjects.
- **WiFiSL Evaluation –** The WiFiSL has been evaluated by users both during at the end of the software development lifecycle. User evaluation results are provided here in a set format. The results include questionnaires, observations and comments from interviews.
- **Converting from Second Life to Open Simulator –** This section details the steps taken to transfer assets such as *Primitives* and *Scripts* from the Second Life island Minerva, into the locally hosted Open Simulator island Aeolus.
- **WiFiOS Evaluation –** Another round of evaluation with a more detailed worksheet is set out. The students use the island Aeolus which is a locally hosted Open Simulator installation.
- **Implementation Issues and Limitations –** Virtual worlds provide a range of advantages and a novel environment in which to explore. The previous sections of this chapter have discussed

the implementations of protocols and designs. This section discusses the limitations imposed on virtual world developers and, where appropriate, suggests ways in which these can be overcome.

- **Reflections on MUVE Suitability –** The previous sections have detailed an implementation of an educational wireless networking laboratory in a virtual world. Here, the suitability of MUVE environment for education is considered.
- **Future work –** The use of a MUVE provides many opportunities; WiFiSL and WiFiOS have shown some of what is possible. This section looks at possible extensions to the current work.
- **Conclusion –** A summary of the work and the main arguments of the chapter are highlighted here.

6.1 *Second Life and Open Simulator*

Figure 60: Screenshot of the Second Life viewer

The two candidates for a MUVE were Open Simulator [216] and Second Life [269]. Open Simulator is an open source implementation of a virtual world allowing individual hardware configuration and management for controlling the virtual world. As such, one can create snapshots of state and can modify the source to a project's specification. A problem with using the Open Simulator platform is that it is still under continuous low-level development. As Open Simulator was under low-level development at the time of this work, it was not considered a stable platform for a virtual laboratory.

Second Life is a mature virtual world developed by Linden Research, Inc. in 2003. Access is free and available through downloadable software from the main website [270], a screenshot of which is shown in Figure 60. Users are referred to as *residents* and can edit or design their own *avatar*'s features from dress to body types. The *in-world assets* are derived from user-generated content. There is an active user base (those who have logged on in the past 60 days) of 1.4 million *residents* and an estimated total number of 15 million [271] (total number of accounts registered with Linden Labs) . Second Life users

can construct grouped objects containing multiple primitive building blocks (*prims*) with custom textures, sounds and actions. This is in contrast to other Massively Multiplayer Online games (MMO) such as World of Warcraft [201] where users pre-install a designed environment from a distributed source such as a DVD.

Second Life is a construction of grouped simulators, each representing an island that can be purchased. These simulators are connected in a two dimensional way allowing a world map of islands to be constructed. One simulator is used per island and records the location of objects, their type and contents, the physics of that world and also the graphics required to display the world in the form of textures, which are cached and delivered to the user. The client software contains multiple standard textures, reducing the load placed on servers by clients connecting for the first time. Each object in the virtual world can be referred to with a 128 bit unique key. The simulator has a limit on the number of *prims* that can exist on the island at one time, currently at 15,000. In [272] the specifications for the machines that host the simulators were detailed as a 64 bit AMD and Intel Xenon, Debian [273] and Linux Apache MySQL and PHP (LAMP) [274-275]. These hosts accommodate four simulators per machine. The physics engine used is Havok 2.0, which performs all collision and dynamics calculations.

Second Life has an economy trading in Linden Dollars ($LD). Trading allows players to buy land, services or items with real money converted to $LD. Second Life has encountered serious problems such as a banking scam [276] that forced the shutdown of the *in-world* banking system. There has also been substantial discussion concerning the protection of children in the online world, prompting an alternative online world to be created that caters just for teenagers [277].

An *in-world* scripting language, Linden Scripting Language (LSL) is provided. LSL development has its nuances and limitations in the importing and exporting of data, especially for real time applications, which is detailed in section 4.11.2. Efficient scripts are required for high performance programming such as the display of a simulator results in real time.

6.2 *Design Considerations*

The software detailed in this chapter seeks to exploit the inherent dynamic nature of a MUVE with a new version of the WiFi Virtual Laboratory. The system must support multiple students in a collaborative environment, creating scenarios fully within a MUVE. The students are provided with networking components in their *inventory*. These networking components can then be deployed throughout the allocated land in the virtual world. Once deployed into the world, they have a physical manifestation and are designed to appear as they would in real life, e.g. a laptop design or a consumer broadband router. Multiple components can be deployed and are visible to all other students on the same island. A mechanism for signalling a traffic connection between two endpoint nodes is used, allowing an immersive and collaborative scenario construction. Scenario construction should be easily

learned and will involve students moving components around in the virtual world and performing actions on components based on a sequence of touches.

Users of the system should be able to alter core parameters of a wireless scenario. The Medium Access Control (MAC) type, routing protocol and signal strength are editable in an easy to use way that involves no knowledge of the scripting environment in which the virtual laboratory is instantiated. Access to such parameters enables greater exploration of the capabilities of the simulator.

Once a scenario has been constructed, on the signal from the user, a controlling object on the island enumerates all the networking components' positions and any traffic connections. These details are then transcribed into dimensions suitable for external simulation. The result from simulating the networking scenario constructed is returned into the virtual world. The virtual world is used to display the networking protocol interactions utilising its inherently immersive visual nature. The display of the results from the simulator is in the form of a series of events. Each event that occurs on the network is animated. All *avatar*s on the same island as the networking components view the resulting display. *Avatar*s use the 3-dimensional ability of the MUVE to view the signals from any distance or perspective.

The system should support several forms of scenario construction, so a student that uses any of the previous versions (WiFiVL I and WiFiVL II) can review that scenario in the virtual world.

6.3 *Educational Scenarios*

This section details the educational areas and scenarios that users of the WiFiSL may choose to focus on. After the description of the system, an example scenario construction is provided. Whilst the laboratory has been focused on the IEEE 802.11 protocols, the jOBA (5.2.3) has flexible interfaces which facilitate future expansions. As such, the scenarios below build on previous worksheets provided to students for the WiFiVL I and II laboratory sessions.

Simulation of ad-hoc WiFi networking

WiFiVL I and WiFiVL II focused on the IEEE 802.11 MAC level exchanges between nodes in a network. Students were required to show understanding of an aspect of a protocol through reviewing their interactions in the WiFiVL player. As the functionality of jOBA was expanded, more scenarios became available for exploration. The exposition of alternative MAC level protocols such as Aloha and Time Division Multiple Access (TDMA) expand the field of exploration in wireless scenarios.

In the MUVE students can place networking components at locations of their choosing and construct traffic connections between them. Signal strengths of the nodes can be altered and this enables the specification of particular wireless scenarios such as the hidden and exposed nodes. The RTS/CTS exchange can be shown in the virtual world, matching the functionality of WiFiVL I & II.

As in previous versions of the WiFiVL, the RTS threshold can be altered by users. This allows the toggling of the RTS/CTS mechanism in IEEE 802.11. A student may construct two scenarios where a

pair of networking components in a traffic connection utilise the RTS/CTS exchange, and then another without its use. A visual difference between the utilisation of MAC exchanges and transport level usages will be apparent. The student, as in previous versions, can reference these complementary scenarios in a lab report when specifying the advantages and disadvantages of RTS/CTS.

Routing protocols

The ability for nodes to construct a series of efficient paths to route information between each other was of great interest in previous practical laboratory sessions. Students were drawn visually to separate nodes attempting to transfer data between each other and the routing protocols that would enable efficient transfer. Routing protocols can be visualised to help with comprehension. The exposition of alternative protocols in jOBA increases the range of simulations available for exploration.

As students can specify the signal strength between nodes, a scenario can be constructed wherein the use of a suitable routing protocol is required. Several nodes can be deployed in the virtual world and a traffic link specified between two out-of-range nodes but where other nodes exist between the two endpoints. Alternative routing protocols (see section 5.2.3) can be used and their effect on MAC usage and efficiency and the interactions can be explored visually. The difference between the use of a flooding algorithm and a shortest path first algorithm exploits the visual nature of the environment as there would be noticeable differences in network traffic and direction.

Another scenario is the removal of a node after a routing protocol centred around that that node has been constructed. The group of nodes would need to reorganise and provide an interesting visualisation to the user. These exploratory experiments by students can help stimulate thoughts on the requirements of efficiency and reliability in routing protocols.

Each of the examples provided above highlight a specific area of interest that can be explored using the system. The combination of aspects highlighted by students will provide interesting and stimulating results for learner and teacher. Whilst the previous examples are expected areas of exploration, users often construct unintended and exciting scenarios.

6.4 *Architecture*

This section details the process a user takes to construct and simulate a scenario in a virtual world to support the earlier design goals for education and interaction. From this process, each of the encompassed components is detailed in a systematic way providing its purpose, representation *in-world*, input and output and its structure. Figure 61 details the components, their interactions and a numerical guide to the process from a user's perspective.

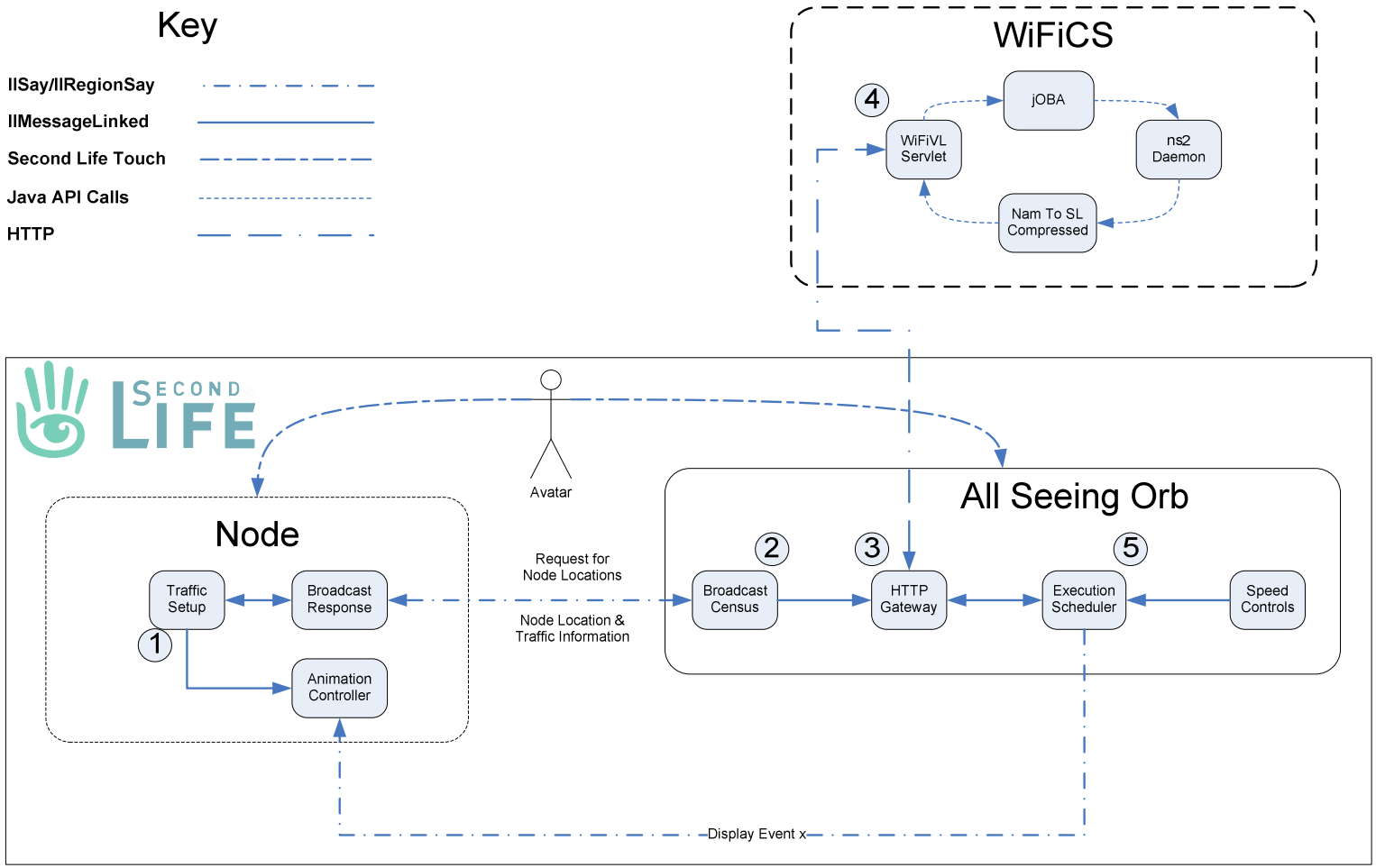


Figure 61: Architectural signal-sequence diagram of the component interaction in WiFiSL

6.4.1 Node

A node is a networking component which is placed in a user's *inventory* and is shown in Figure 62. The virtual world representation is that of a modern day Macintosh laptop and is extended from an open source creation [278]. The components' interactions are through touch only and the details of the interaction methodology are provided in section 6.6. The output from the node is three-fold:

1. it provides a visual reference to the node location in the virtual environment as deployed by the user
2. it displays the results of the animation after it has been simulated in ns-2
3. it provides visual confirmation that the user has constructed a traffic link between two nodes

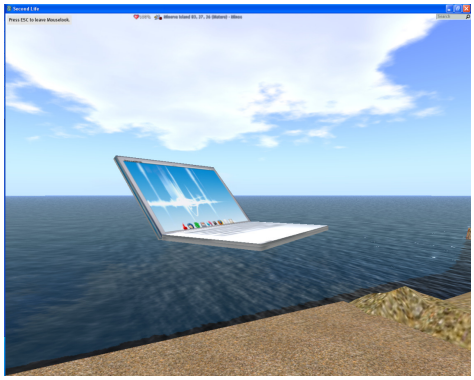


Figure 62: Screenshot of a networking component in Second Life

Representation	Model of a laptop (see Figure 62)
Purpose	Provide a visual reference for the location of the node in a constructed scenario
Method of Interaction	Touch
Input	Touch from the user, signals from other nodes on the island on a pre-assigned communication channel
Output	Animation of signals

Table 18: Summary of the key elements involved in the Node component

6.4.2 All Seeing Orb

The "All Seeing Orb" is represented in the virtual world as a sphere shown in Figure 63. The All Seeing Orb contains the scripts required for eliciting Node configuration and gateway management. The All Seeing Orb acts as:

- a central point for configuration aggregation – details of a scenario for execution; this attribute is detailed as the Broadcast Census in 6.7.2.
- a translation of *in-world* information such as Second Life keys into identifiers used in ns-2 and vice-versa.

- an invocation of the WiFiCS which supports WiFiSL through HTTP as described in the Gateway section 6.7.3.
- the timely execution of events which result from the simulation as described in the Execution Scheduler section 6.7.4

The input is the location, identity and traffic connection details of all nodes on the island that are involved in the simulation. Through the WiFiCS, the output is a series of commands sent through a channel causing the networking components to animate certain network events.

Representation	Spherical object (see Figure 63)
Purpose	Provide a central controlling point for the initiation of the simulation of the scenario, the request for the simulation to be completed and then instructing networking components to execute resulting events at the correct playback time.
Input	Console or touch commands, location of networking components on the island and any traffic connections between them.
Output	Broadcast requests to networking components, HTTP requests to the WiFiCS and instructions on which signal to animate at which particular networking component.

Table 19: Summary of the key elements involved in the All Seeing Orb component

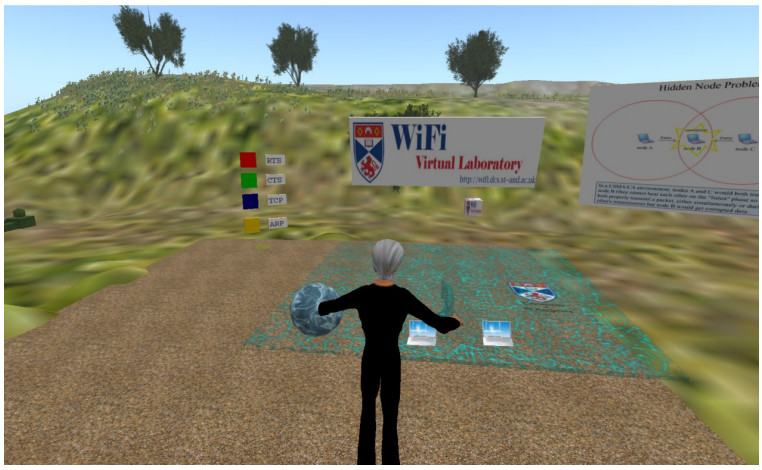


Figure 63: Screenshot of the WiFiSL All Seeing Orb attached to the *avatar*'s left hand

6.4.3 WiFi Constructor and Simulator

The WiFi Constructor and Simulator (WiFiCS) component in Figure 61 represents a part of several components as described in section 5. These reusable components are server-side and contain methods for constructing a representation of a scenario in Java. This representation is translated, using the Java OTcl Build API (jOBA) into an OTcl file that can be simulated in ns-2. WiFiCS compresses the discrete event results from the simulator into a more space efficient format (see section 6.7.3).

6.4.4 Process Description

A scenario refers to a user's construction of networking components, the traffic interactions between them and advanced parameter settings. The steps a user takes to deploy a network, construct a scenario and review the results were shown in Figure 61 with a textual description below.

A user deploys networking components into the virtual environment from their *inventory*. Connections are created by touching the networking components. Touching two in succession creates a connection, the order of which denotes the sender and receiver. Visual confirmation is animated to the user. The confirmation animation is automatically stopped after 4 seconds. The connection created by the user is recorded as a file transfer between the two networking components. The start time for the transfer is assigned by the orb. This removes the necessity for the user to specify valid start and finish times during scenario construction.

The gateway broadcasts a request to all networking components to respond with traffic configurations and position reporting. Each node is identified by its 128 bit key assigned by Second Life on first creation (a process known as *rezzing* [279]). A traffic link consists of sender and receiver keys, which is then translated into a numbering system beginning at 0 and going to n-1 (where n is the number of nodes). This translation and storage is important as the key is used for specifying the destination of animation of network events. The traffic descriptions also contain start and finish times specified by the networking components' wall clock time value.

Once all the information has been collated, a special URL is constructed and submitted to the system used by both WiFiVL and WiFiVL II (WiFiCS). This URL can be taken and used in any of the other virtual laboratories created (WiFiVL I & WiFiVL II).

The WiFiCS translates the URL into a description and then executes the simulation, returning the results to the orb in a compressed format detailing every network event.

The orb controls the animation of events in the virtual world with an execution scheduler. Nodes are instructed to perform animations by the scheduler on a pre-assigned channel.

6.5 *Visualisation Considerations for Wireless Transmissions*

In a real-world connection between two wireless points, these signals are invisible. Another feature of a real world connection is that the speed of communication is too fast to be observed. An important design consideration was the visualisation of a wireless transmission in a 3-dimensional environment. This is an area that has not received much in the way of user research. While textbooks will commonly use a circle or a lightning icon to represent a transmission, the virtual world allows for a richer representation. WiFi operates using radio waves, which are the modulation of electromagnetic fields.

Ideally the features that would be useful to observe in a wireless experiment are:

- **Progression**: The interface design should consider how packets are transmitted and their propagation through the medium.
- **Addressing Information**: A packet sent out in a real world wireless deployment may have an intended recipient. The recipient may be a specific node with a certain address or it may be intended for all nodes in a certain geographic area.
- **Functionality**: Each protocol data unit transmitted by a node has a specific purpose. The interface design should present a way of immediately determining signals' intended functionality, e.g. whether the unit transmitted is data or a signalling frame.

These three properties define the key requirements in creating the observation of a simulation which is useful and interesting. One way of representing a wireless packet is as an expanding sphere, which can be seen in Figure 64. An advantage to this representation is that it shows the range at which the signal can be received by another node. What is not shown is the intended recipient.

Another approach to the modelling of a wireless signal is through the use of particles being emitted from the transmitting node. The *llParticleSystem* [280] is used to create particles in Second Life. Each script has access to only one particle system. By using the particle emitter provided in the API, instead of a proprietary implementation, all effects are performed on the client side only. The use of client side rendering ensures no extra resources are put on the island's simulator. Client-side rendering is used for cosmetic effects during time-critical online games, thus bandwidth can be allocated to the more important information. The format of the API command is *llParticleSystem(list parameters)*.

The parameters available to set are:

- `PSYS_PART_START_COLOR` – Initial colour of the particle, specified as <red,green,blue> tuple
- `PSYS_PART_END_COLOR` – Final colour of the particle, specified as a <red, green, blue> tuple
- `PSYS_PART_MAX_AGE` – A value, in seconds for the life of a particle
- `PSYS_SRC_ACCEL` – Determines the particle's acceleration
- `PSYS_SRC_BURST_PART_COUNT` – Number of particles to emit in a burst
- `PSYS_SRC_TARGET_KEY` – Particle's movement towards a specified Second Life key

The effects of the particle system, if no maximum age is specified, will remain until the *llParticleSystem* command is invoked with an empty array.

Particles could be used to represent a quantum of energy being emitted by the transmitter. An example of this can be seen in Figure 65. In this approach, whenever a node is transmitting a signal a colour-coded explosion of particles from the senders' antenna is emitted. As the particles expand they become increasingly transparent, fading away. A disadvantage to this approach is that it may wrongly imply to the student that multiple emissions are transmitted.

An alternative approach to the problem of representing wireless signals is shown in Figure 66: a group of particles moving towards a destination. The frame that is currently being transmitted can be colour coded to represent the functionality of the frame. The disadvantage of the directed transmission is that it hides the broadcast nature of wireless. This broadcast nature is important for understanding when the IEEE 802.11 Network Allocation Vector (NAV) timers are updated, for example. As such, it is not immediately clear from observation that all nodes in a certain area will receive the signal and not just the intended recipient. These requirements and proposed solutions are summarised in Table 20.

	Propagation	Intended Recipient	Functionality
Beam	✓	✓	✓
Expanding Sphere	✓	×	✓
Particle Emission	✓	×	✓

Table 20: Summary of the features of alternative approaches to representing a wireless signal

The final design choice was to use a combination of the above. Expanding spheres, as shown in Figure 64, for packets have a broadcast address such as the RTS/CTS and routing packets. Packets that have a destination address are shown as being directed, i.e. the particles flow from a source to one destination, as shown in Figure 66.

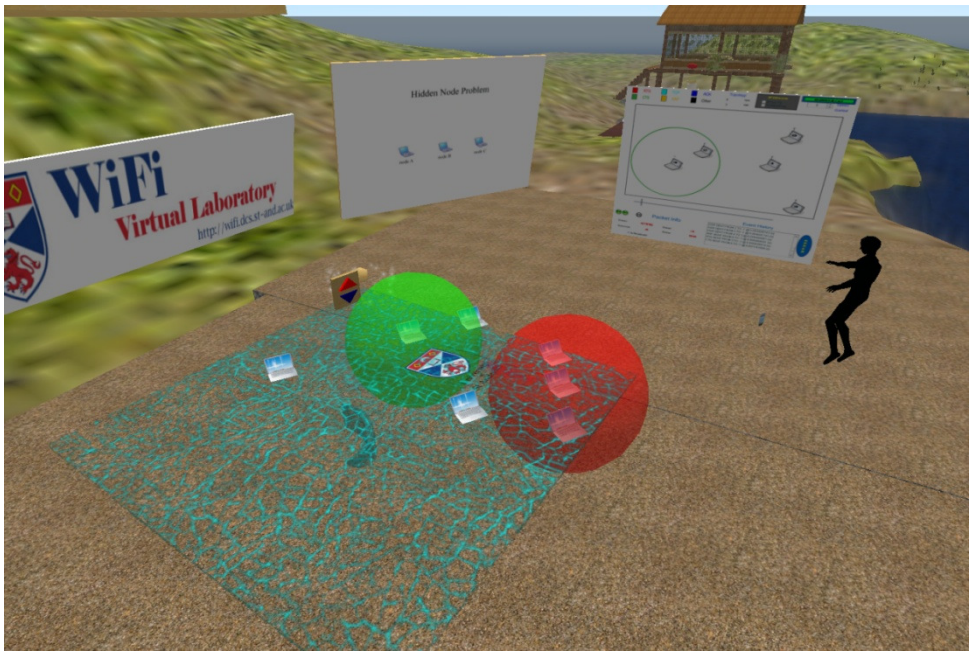


Figure 64: Screenshot from Second Life of two wireless signals that have expanded to a set size with a transparent setting during a simulation animation

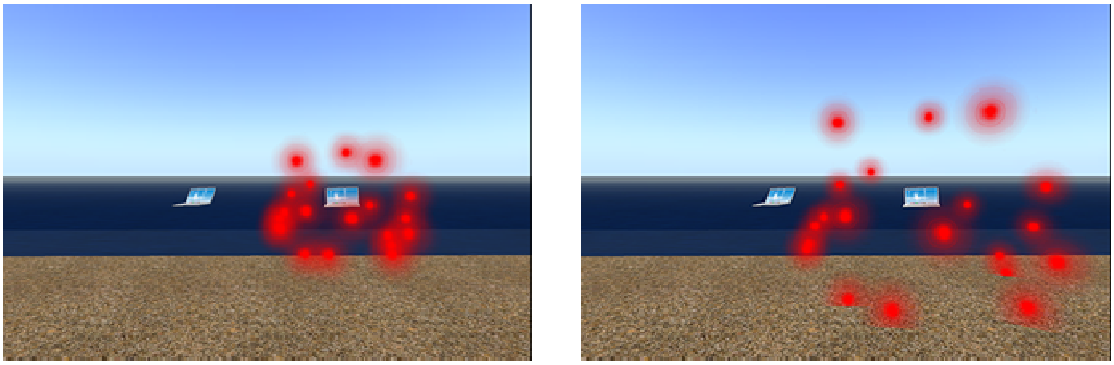


Figure 65: Screenshot from Second Life showing an explosion of particles from a central point

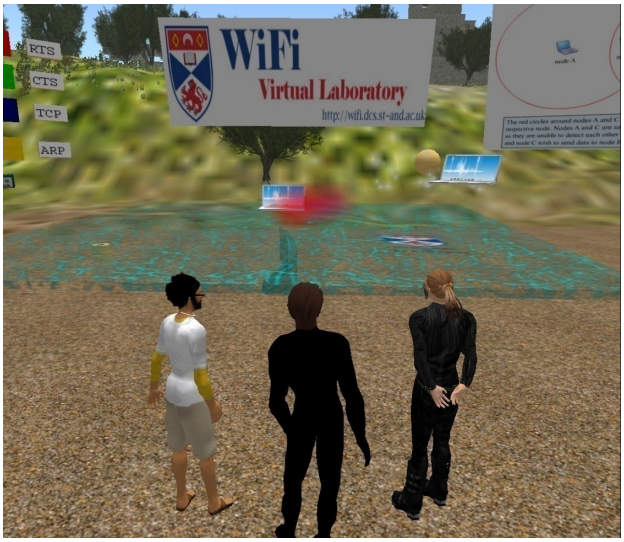


Figure 66: Screenshot from Second Life showing a representation animation of a wireless signal propagating from one network component to another

6.6 *User Interface Design*

This section details the design that allows users to create a scenario in the virtual world. A scenario construction involves traffic between nodes. Traffic may be of different types such as FTP, CBR or HTTP. The construction of traffic between nodes should be intuitive and accessible through an immersive 3-dimensional environment.

Multiple networking components can be placed in the environment, as shown in Figure 67. The use of the laptop model is to make the simulation as immersive as possible and also to enable a quick comprehension of a scenario. Previous laboratories have used command line arguments, configuration files, HTML forms and point and click 2-dimensional configuration. The ability to place nodes at any location on the island greatly increases the flexibility and possibilities of the laboratory.

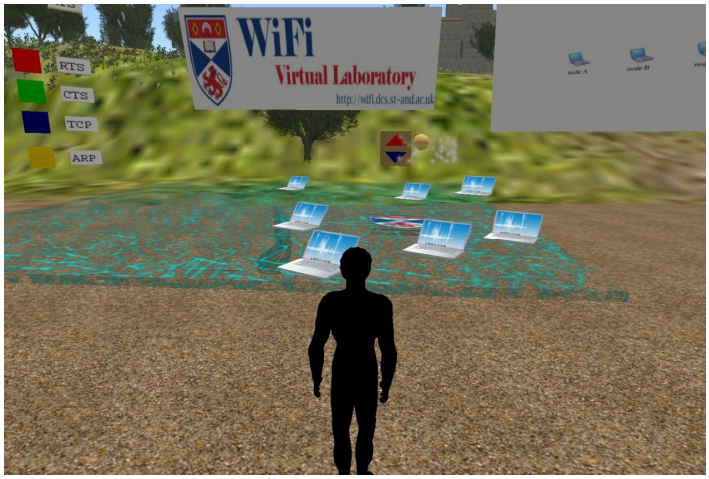


Figure 67: Screenshot of several wireless networking components in Second Life

Once users have placed the networking components on the island, traffic connections must be made between two endpoints. The traffic connection must be created by the user and recorded by the system. Connection details such as duration and identities of the two endpoints are then used to construct the URL that describes the scenario for simulation by ns-2 through the WiFiCS.

There are many considerations when designing a 3-dimensional interface. A challenge for the design was to enable a fully immersive traffic construction system in which no text files were needed. As such a system has been designed that allows an *avatar* to indicate a traffic connection between two endpoints by touching the networking components in order. By touching one node followed by a second node inside a certain time period, a traffic connection is created. The two nodes stay connected, transferring information until the user double clicks on the sending node. Each example construction makes reference to a signal sequence diagram unique to that scenario and shows the user's interaction with the objects through touch and the internal state transitions of the objects.

Since the networking components are altered into different states based on the interactions of both users and other networking components, it is possible to represent the components using a finite state machine description, as shown in Figure 68. A finite state machine enables a formal representation of all the states a networking component can be put in and the triggers for each state transition.

The following is a brief description of each of the states detailed in Figure 68, as shown in the key. Each square is a final state and not time-dependant. Each circle is a time-dependant transition state and as such the node cannot remain in that state indefinitely.

- **Listening** – Listening is the initial state of all the networking components when first deployed. This is also the state to which they return when not transmitting.
- **Touched** – Whenever a networking component is touched, a message is broadcast to all other nodes. The component that was touched is then placed into the Touched state. From this state, the

component may become a transmitter or, if a certain time period expires, it will return to its initial Listening state.

- **Awaiting Connection (Initial State)** – On receiving indication that another node has been touched the component will go into the Awaiting Connection state to indicate that it may be touched next and will be the designated receiver in a traffic connection. If the networking component is not touched within a set time period or another node is touched, then the component will return to its Listening state. If it is touched then the component enters the Receiving state.
- **Transmitting** – This state is reached through being the first node touched, then a second component on the island is touched which causes the first component to become the transmitter. The transmitter has the duty of recording the duration of a traffic connection and the destination. The transmitter is also required to animate the connection established between the two components. Once double clicked, the Transmitting component will return to its Listening state.
- **Receiving** – A networking component will reach this state if another component is touched first, then this component is touched. When in the receiving state, the component is not required to animate or record any details of the communication. As such, it can return to the Listening state safely. The receiving component signals to other networking components that it is the receiver, allowing other components to reset to the Listening state.

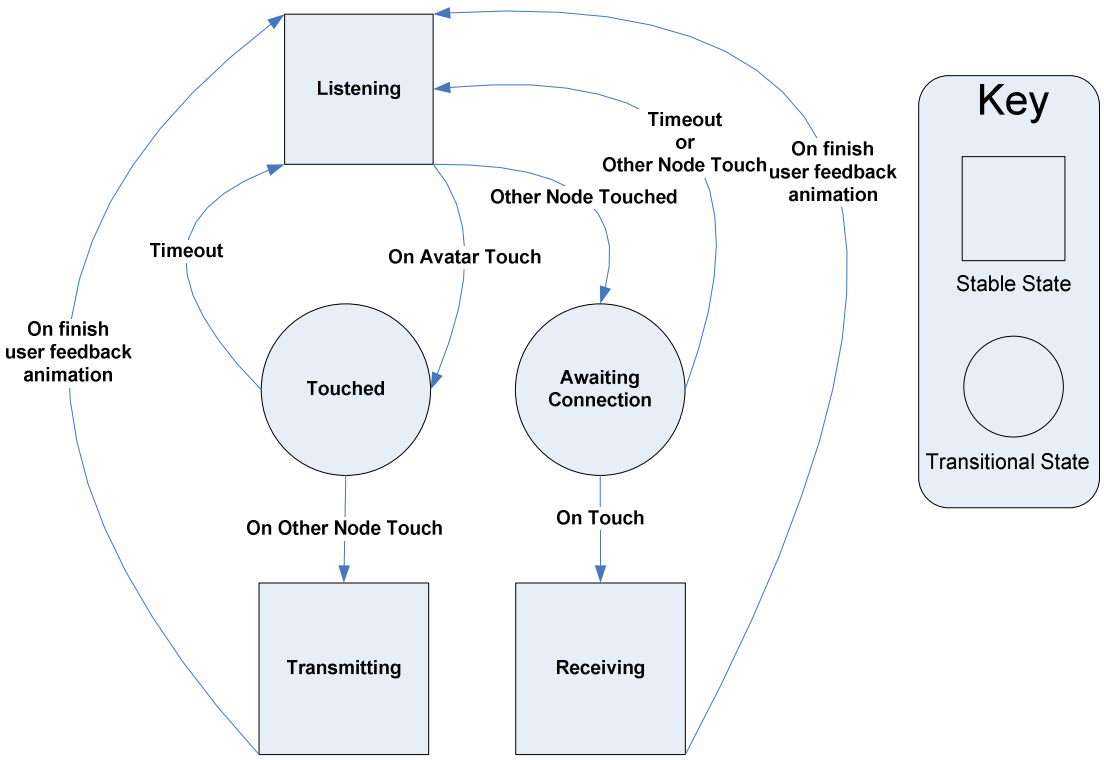


Figure 68: State transitions diagram representing the changes in a node due to user touch events

A traffic connection between two nodes necessitates a sender and receiver of data. The user initiates a traffic connection by touching Node A ($\mathbf{t_1}$, Figure 69), which causes that node to announce on a channel it has been touched. This is shown in Figure 68 as a state transition from "Listening" to "Touched". All other nodes in the region will change state into "Awaiting Connection". The next node touched inside a set time frame will be designated as the receiver. In Figure 69, $\mathbf{t_2}$ will trigger Node B to change state from "Awaiting Connection" to receiving, as shown in Figure 68. When Node B is touched at $\mathbf{t_2}$, this event change is announced to all nodes in the region, allowing all other nodes to change their state back to the initial state of "Listening". Only Node A stays in the transmitting state as it has to record the details of the traffic connection, unlike the receiver Node B. Therefore all other nodes in the island that were put into "Awaiting Connection" will change state back to "Listening".

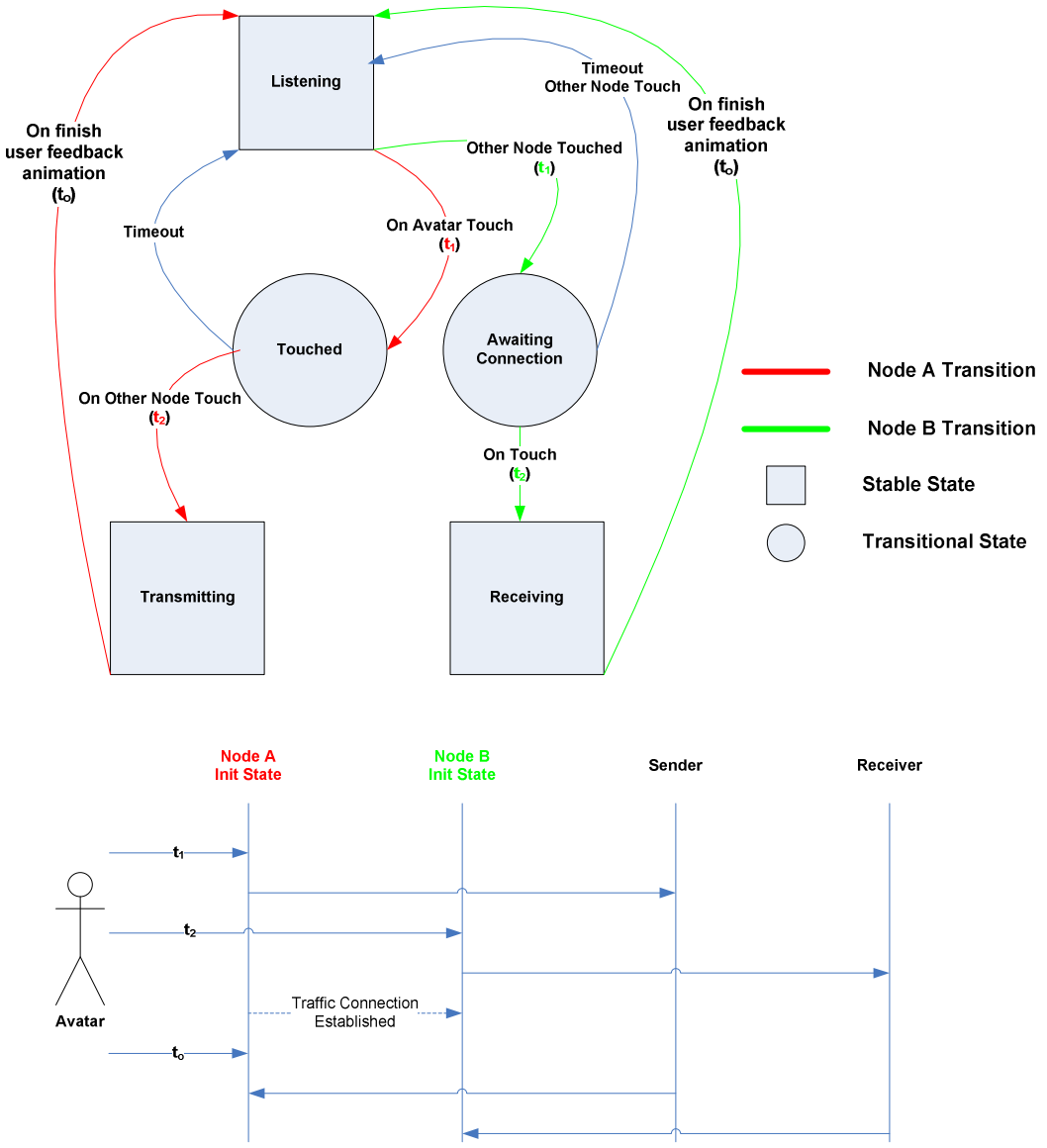


Figure 69: Signal sequence diagram with a matching state transition machine detailing the actions a user performs to construct a traffic scenario in the WiFiSL and the state changes

The connection is terminated automatically using a timeout event as shown in Figure 69 by $\mathbf{t_o}$. This resets the state in Figure 68 from "Transmitting" to "Listening". When a broadcast message requesting a nodes status is transmitted, the following information is returned by:

Node A

- The Second Life unique identifier announced by Node B at point $\mathbf{t_2}$ in Figure 69
- Details of any traffic connections where the node is the transmitter
- Its location on the island, given as a vector specifying x, y and z coordinates

Node B:

- Second Life unique identifier
- Its location on the island, given as a vector specifying x, y and z coordinates

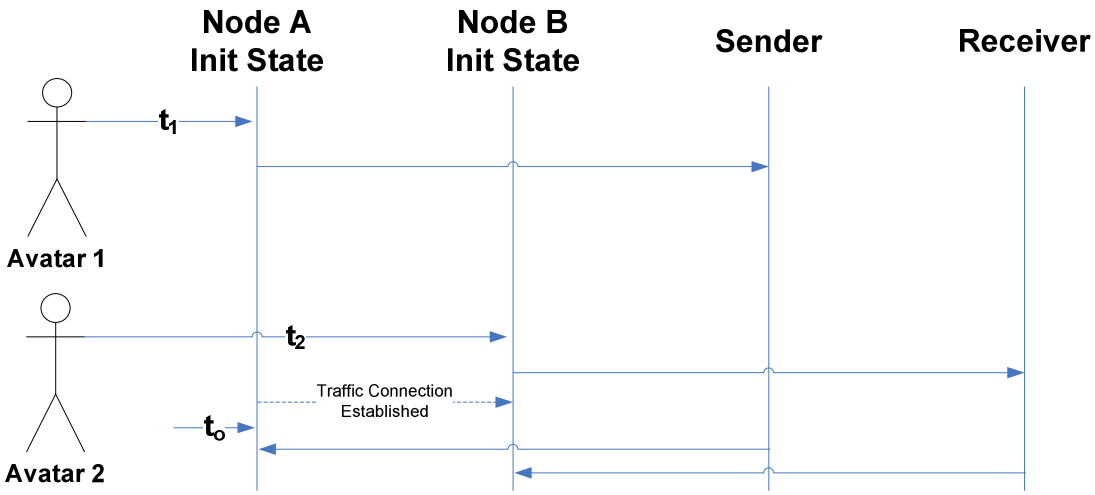


Figure 70: Signal sequence diagram showing the signals used during a collaborative scenario traffic construction

Figure 70 shows the same resulting traffic connection between two nodes as Figure 69 but the connection is completed through a collaboration (for more information on the scope of each *avatar*'s scenario construction see section 6.6.2). *Avatar* 1 puts Node A into its sending state followed by *Avatar* 2 putting Node B into receiving state. This example of a collaborative scenario construction demonstrates the capabilities of the MUVE in which WiFiSL is situated.

A user indicates the order of file transfers through the order that they make connections between nodes. When a connection is created between two nodes, the user is shown the state of the system (transmitting) by displaying particles being sent from the sender to receiver.

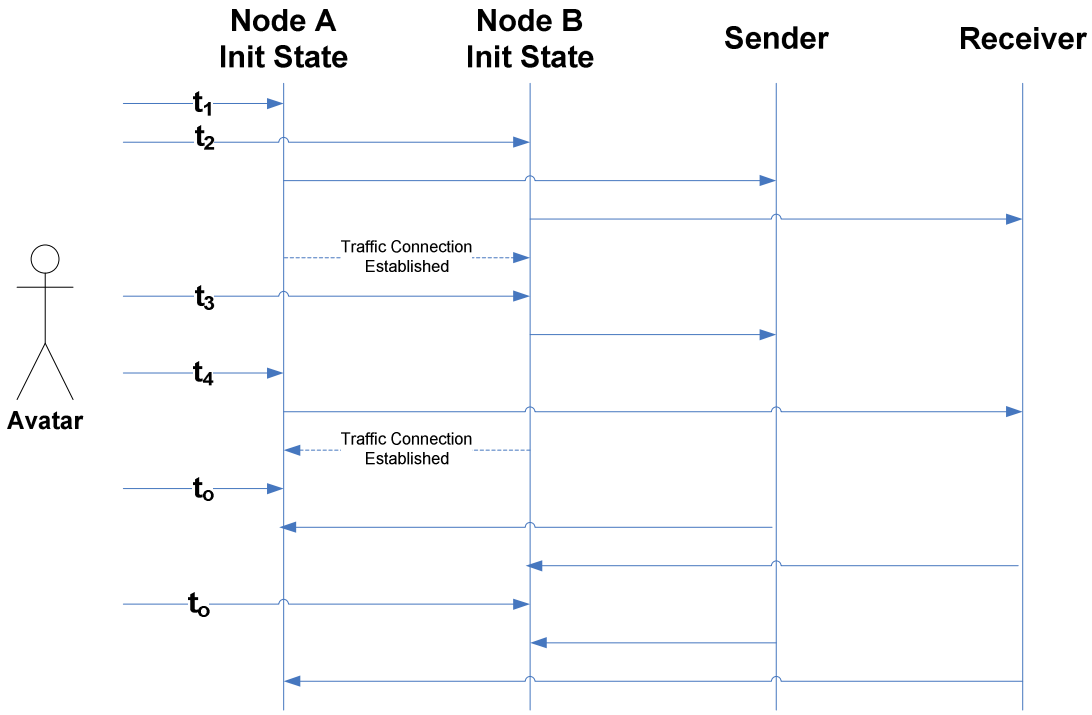


Figure 71: A signal sequence diagram showing the construction of two concurrent traffic communications between Node A and Node B

Figure 71 shows two nodes where Node A and Node B are sender and receiver in a traffic connection. Touches t_1 and t_2 establish a connection where Node A is the sender and Node B is the receiver, as indicated by the signal "Traffic Connection Established" between the two. The touches t_3 to t_4 cause a traffic connection between the two again, but this time Node A is the receiver and Node B is the sender. The timeouts on the sender will cause the sender to stop transmitting and record the information about the connection, such as duration, start time and destination. The timeout is a period in which a display is shown to the user confirming they have constructed a traffic link successfully.

Sending particles to a destination node is a design requirement in WiFiSL. This is used to indicate a communication link between two nodes in the network. Extensive use of the llParticleSystem [223] was utilised, allowing the specification of a target for a set of particles. The particles' start and end colours are custom made depending on the type of packet. For the scenario construction a simple blue constant particle emitter was used. For the scenario playback, i.e. the results of the simulation, if the packet had a broadcast destination then an expanding sphere would be used. To do this an expanding spherical primitive was designed and added to the *inventory* of the node then programmatically *rez*zed [279], i.e. created or instantiated in Second Life environment. The sphere accepts a number as an argument, that number designates the appropriate colour code to use. Once *rez*zed the object expands up to a certain size then automatically removes itself from the world. If the packet has a destination then the *llParticleSystem* target provision would be used to guide the particles. The particles are also

be appropriately coloured. This differentiation between addressed and broadcast packets allows various aspects of the WiFi network to be instantly recognisable.

6.6.1 Console Interface for Power Users

In WiFiVL I students used a HTML form which provided a quick way of generating a scenario and exposure to detailed settings. The 3-dimensional environment should be all encompassing and should allow the student full control over scenario construction. A HTML form does not translate well into a virtual environment. Initial experiments involved the users interfacing with a dialog system using the chat commands. The dialog system provides commands to review the state of the scenario construction, helpful instructions for specifying communications between deployed nodes and commands to fetch the result of the scenario construction. In keeping with usability principles, an amalgamation of the two will provide power users with shortcuts (section 4.2). It is also important to build up the confidence of users so they will develop and become power users of the system. A text system overcomes the problems of trying to touch two objects inside a set time frame that are separated by an entire island where the default draw distance for objects in Second Life is approximately 64 metres.

```
[1:30] Minos Deerhunter: help
[1:30] OraclelistenForConfiguration:
*******************************
Welcome to the WiFiVL, say the command to execute.

Listnodes                                    shows nodes in the field

tlink [srcId destId startTime finishTime]    sets up a link between two nodes e.g.
                                             tlink 0 1 0 2 – traffic link between
                                             node zero and node one starting at time
                                             0 and finish at 2(s)

wifireset                                    Resets all traffic links

listnodes                                    Lists the known nodes

wifirun                                      Runs the simulation
*******************************

[1:31] Minos Deerhunter: tlink 0 1 0 1

[1:31] OraclelistenForConfiguration: Traffic list:&tt=0,1,0,1,FTP,TCP
```

Figure 72: Example use of the text interface to WiFiSL for power users

6.6.2 Scope of Simulations

This section details the scope of a scenario construction in the virtual world. Scope is used to describe the number of *avatar*s and networking components that can influence scenario construction. In a collaborative environment, the scope of the components includes all participating *avatar*s. Where there are multiple experiments on the same island then the scope should be limited to the participants in each simulation.

A communication channel is used by scripts to send information to other scripts or *avatar*s. There is a wide range of channels to choose from and there is an option to define the range in which a message

will be transmitted. When deploying the WiFiSL tool there are two important communication channels:

- Broadcast Census Channel and Animation Channel – The Broadcast Census channel is used to find out the location of all nodes on the island and any traffic links that may have been constructed. The nodes reply with this information on the same channel it was requested. The Animation Channel is used to trigger an animation event in a node. The trigger is controlled by the Execution Scheduler specifying the Second Life ID of the node and the type of signal it should animate.
- Inter-networking component communication – When constructing a scenario through touching nodes on the island, each of the nodes communicate their state using this channel.

All components must use the same communication channel, as must the central controlling orb. If multiple students use the same communication channel then each student's actions will affect every other networking component on the Island. This allows for collaboration but can also introduce difficulties when there are many students *in-world* attempting to manipulate a networking component. This system allows for multiple independent scenario construction through the altering of the communication channel which is stored at the top of the scripting file in the networking component. There remains the option to have a collaborative construction through the use of *avatar*s' touching nodes.

6.6.3 Traffic Implementation

This section describes the method in which the nodes signal to each other that a traffic connection has been initiated by the user. This is the implementation of the design discussed in section 6.5.

Figure 73 shows the code structure of the networking component. Protocol specific string messages (TOUCHED_MESSAGE, urlParameterRequest) are also declared in the scripts header for inter-node communication when it has been touched or when instructed to broadcast its traffic and location information. The script "Effects" in Figure 73 is used for creating effects that animate the results of the simulation to the user. This animation is achieved through extensive use of the llParticleSystem [280].

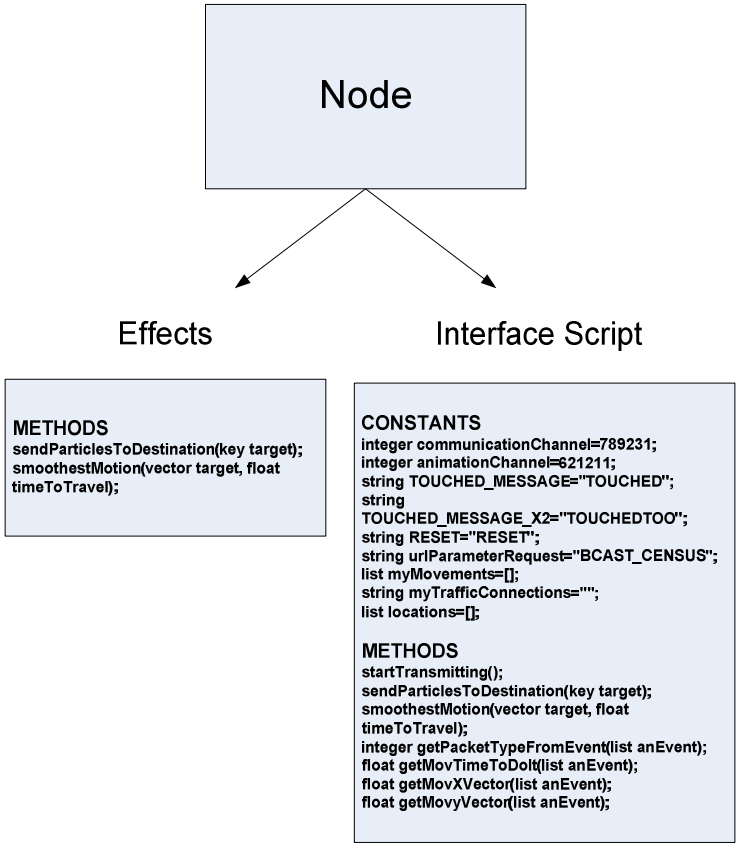


Figure 73: LSL construction of a networking component in WiFiSL - key aspects of code used and interface definitions

When an *avatar* touches two nodes inside a set time period, it creates a traffic link between the two nodes. The signal sequence diagrams in Figure 69, Figure 70 and Figure 71 provide information on the way the *avatar* interacts with the nodes. This section will provide more detail on the protocol developed to support such interactions and makes extensive use of the state machine illustrated in Figure 68.

When a node is touched by an *avatar*, a message is transmitted on a channel in Second Life. Both the message and channel number are shown in Figure 73 under the constants "TOUCHED_MESSAGE" and "communicationChannel". All nodes on the island listen to the communicationChannel for messages and once a "TOUCHED_MESSAGE" is received they enter the state *Awaiting Connection*. If a node is touched whilst in the Awaiting Connection state, then it is clear that that node is the intended receiver of the traffic scenario. When the receiver node is touched, it transmits a message on the communicationChannel, shown in Figure 73 as "TOUCHED_MESSAGE_X2" and this will set all other nodes back to their listening state. This is shown in Figure 68 by the transition from *Awaiting Connection* to *Listening*.

6.7 *Animation Controller*

The Animation Controller is the component responsible for animating packet transmission events and indicating the existence of a current traffic link during a traffic scenario construction. The Animation Controller is triggered by the networking component and by the execution scheduler. The Animation Controller listens on a set channel and whenever it receives a message indicating that an animation is required for a certain packet, it will check to see if the networking component identifier specified matches its own. The identifier used is the Second Life 128 bit key.

When a node is animating a traffic construction, the node must be in the transmitting state. When animating this event, the node transmits a series of particles emanating from it and targeted at the receiving node as shown in Figure 74. During traffic construction there is no simulation so the only representation shown is a plain blue series of particles that travel to the destination node.

The llParticleSystem command provided by LSL enables the generation of temporary particles which are rendered only on the client side. This command can be used for generating single or multiple particles of varied size, colour and lifetime.

Figure 74: Screenshot of the traffic connection between two nodes as created by the *avatar* when constructing a scenario

6.7.1 Broadcast Response

The Broadcast Response component is used to listen for requests for configuration information about the node. The node registers a listener on a defined channel and awaits commands. When the node receives the text stored in the variable urlParameterRequest, here defined as “BCAST_CENSUS”, the node will return its own location information. The node’s unique identifier and any traffic links that have been constructed while it was the sender are transmitted back on the same channel. Only the nodes that were part of the transmitter are required to return traffic information as this reduces duplication of information.

The Broadcast Response component has no interaction with the user and serves only as a reporting mechanism on the current node state. All communication is done through the chat channels as the use of llMessageLinked restricts communications to linked objects whose centres can be at most 5 metres apart from each other.

6.7.2 Broadcast Census

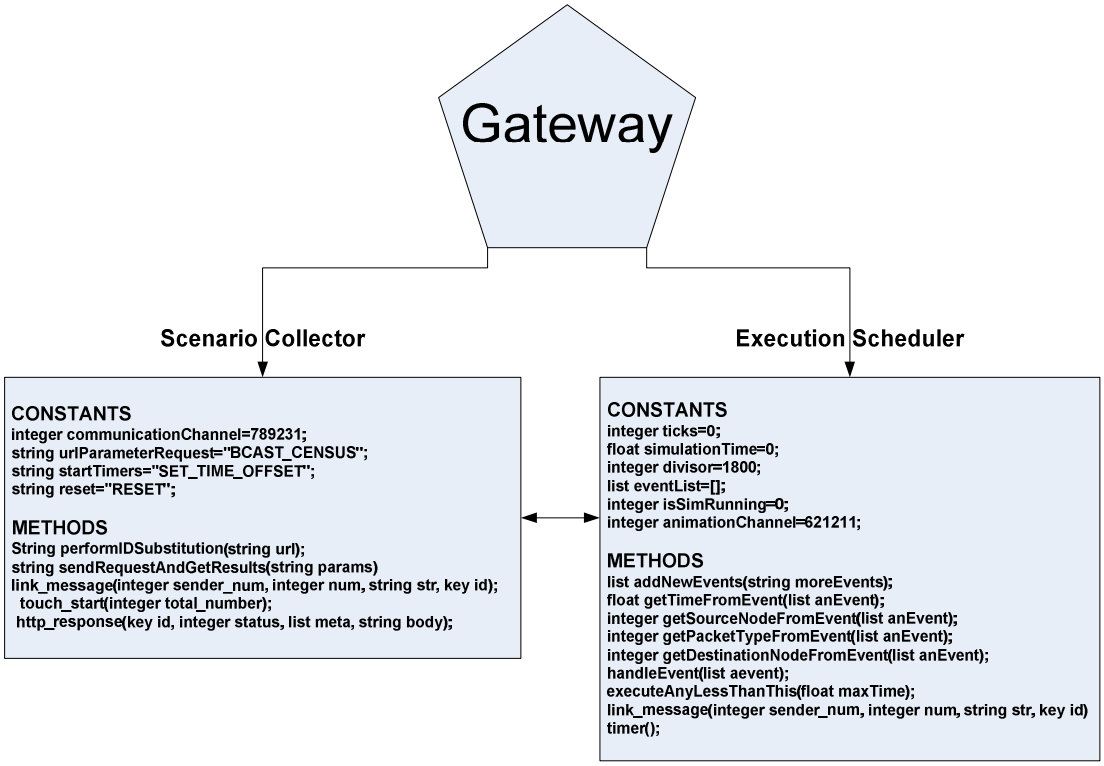


Figure 75: Scripting language structure for constructing a gateway that supports scenario collection and execution scheduling

Figure 75 shows the main gateway comprising two scripts. The "Scenario Collector" is used to gather information from the networking components and is achieved via the communication channel as described in section 6.6.3. Networking components transmit their location, unique identifier and traffic links. The unique ID is a 128 bit Second Life key which the Gateway will translate into the ns-2 notation of 0 to n for identifiers. This is translated back to Second Life keys before animation.

Efficient communication between the Scenario Collector and the Execution Scheduler is required. This is often difficult due to restrictions in Second Life. Whilst the two scripts are contained within the same objects and resident in the same simulator's memory, no method invocation with declared parameters is possible. The communication requirements are both low latency and high bandwidth. For this reason, llLinkedMessage was chosen. For a full discussion on the different communication methods and their limitations see section 0.

6.7.3 HTTP Gateway and Modifications to WiFiCS

The HTTP Gateway manages the HTTP requests from Second Life to the WiFiCS. Each scenario in WiFiVL I and II were described using a URL only and this design choice has been followed through in WiFiSL, as such HTTP requests are required to gather the results of the simulation. The WiFiCS has been extended to facilitate the prohibitive environments of coding in Second Life.

There are strict limitations imposed on the amount of data that can be imported to Second Life. Second Life scripting allows for several ways to call out including Email, HTTP and XML-RPC (see section 4.11.2). The maximum payload of a HTTP call is set at 2048 characters. Both the XML and NAM version of a network trace are too large to import a sufficient number of events to enable a smooth playback of the simulation. This severe limit on the number of characters requires a new way of describing the network simulation in a compact and efficient way. Therefore a new library was developed and added to the WiFiCS to convert NAM into a compressed comma and colon delimited format.

```
+ -t 2.073511123 -s 1 -d 0 -p DSR -e 44 -c 2 -a 0 -i 51 -k RTR
- -t 2.073511123 -s 1 -d 0 -p DSR -e 44 -c 2 -a 0 -i 51 -k RTR
h -t 2.073511123 -s 1 -d 0 -p DSR -e 44 -c 2 -a 0 -i 51 -k RTR
+ -t 2.073986123 -s 1 -d -1 -p ARP -e 80 -c 2 -a 0 -i 0 -k MAC
- -t 2.073986123 -s 1 -d -1 -p ARP -e 80 -c 2 -a 0 -i 0 -k MAC
h -t 2.073986123 -s 1 -d -1 -p ARP -e 80 -c 2 -a 0 -i 0 -k MAC
r -t 2.074626312 -s 0 -d -1 -p ARP -e 28 -c 2 -a 0 -i 0 -k MAC
+ -t 2.074701312 -s 0 -d -1 -p RTS -e 44 -c 2 -a 0 -i 0 -k MAC
- -t 2.074701312 -s 0 -d -1 -p RTS -e 44 -c 2 -a 0 -i 0 -k MAC
h -t 2.074701312 -s 0 -d -1 -p RTS -e 44 -c 2 -a 0 -i 0 -k MAC
r -t 2.075053500 -s 1 -d -1 -p RTS -e 44 -c 2 -a 0 -i 0 -k MAC
+ -t 2.075063500 -s 1 -d -1 -p CTS -e 38 -c 2 -a 0 -i 0 -k MAC
- -t 2.075063500 -s 1 -d -1 -p CTS -e 38 -c 2 -a 0 -i 0 -k MAC
h -t 2.075063500 -s 1 -d -1 -p CTS -e 38 -c 2 -a 0 -i 0 -k MAC
r -t 2.075367689 -s 0 -d -1 -p CTS -e 38 -c 2 -a 0 -i 0 -k MAC
+ -t 2.075377689 -s 0 -d -1 -p ARP -e 80 -c 2 -a 0 -i 0 -k MAC
- -t 2.075377689 -s 0 -d -1 -p ARP -e 80 -c 2 -a 0 -i 0 -k MAC
h -t 2.075377689 -s 0 -d -1 -p ARP -e 80 -c 2 -a 0 -i 0 -k MAC
r -t 2.075793877 -s 1 -d -1 -p ARP -e 28 -c 2 -a 0 -i 0 -k MAC
```

Figure 76: Typical NAM sample, 1202 characters

```
7,1,0,2.072511:2,1,-1,2.073986:0,0,-1,2.074701:1,1,-1,2.075063:2,0,-1,2.075377
```

Figure 77: Compressed scenario trace from NAM, 79 characters

```
integer RTS = 0;
integer CTS = 1;
integer ARP = 2;
integer ack = 3;
integer ACK = 4;
integer TCP = 5;
integer UDP = 6;
integer DSR = 7;
integer MOV=8;
integer unknown = 9;
integer CM8=10;
integer AODV=11;
integer TORA=12;
integer packetTypeIndex=0;
integer sourceNodeIndex=1;
integer destinationNodeIndex=2;
integer timeIndex=3;
```

Figure 78: LSL scripting constants used to access information from the compressed scenario trace in Figure 77

Figure 76 shows a sample output from an ns-2 simulation in the NAM format which details network events. The compression service strips out events that will not be shown to the user such as the enqueue and de-queuing of packets and looks only at events that are transmitted on the network. Each packet type is given a single lower or uppercase alpha or numeric code, allowing for 61 packet types to be described. The time value is shortened, losing two of the least significant values. This loss of precision in the time value does not affect the way the simulation is viewed as the time difference is negligible. Source and destination are put in a pre-defined position and the text and packet size is ignored. A generic description of the protocol is: [Packet_Type],[Source_Node],[Destination Node],[Time Of Event] as is demonstrated in Figure 77. Using this compression method, the network trace in Figure 76 is compressed to 15% of its original size. This compression service was the only addition to the WiFiCS. The construction of scenarios into a URL followed the same structure as WiFiVL I & II.

An additional helper filter was created that would allow indexing into the results of a simulation. Since Second Life was incapable of accepting large HTTP responses and storing these in variables, the starting result number and result size could be specified. This allows polling from the resource limited scripts in Second Life, allowing large simulations to be shown in Second Life.

Due to the small amount of size available per request, the Gateway must perform multiple HTTP requests. If the number of network events available to the Execution Scheduler is less than the threshold value of 10 (line 3), then a linked message is sent to the Scenario Collector (line 6). A flag is set so that when the script receives a HTTP response it will know that a simulation is in progress and that the results are related to the current simulation and are to be appended to the network events string (line 5).

When the Scenario Collector receives the link message, the offset into the simulation results is increased by a set size. This is increased until no more results are available from the WiFiCS Servlet, with the result size set to 25. The modified request is sent to the WiFiCS Servlet and the response is provided as a registered event. Once the results of the HTTP request are received, it is then passed back to the Execution Scheduler using a link message.

```
1.  integer requestingMoreSimResults = 5;
2.  if(noMoreResults!=1&&llGetListLength(eventList)<10){
3.  llOwnerSay("Requesting more results");
4.  noMoreResults = 1;
5.  llMessageLinked(LINK_THIS,requestingMoreSimResults,"",NULL_KEY);
6.  }
```

Figure 79: LSL sample showing the Execution Scheduler and Scenario Collector in the Gateway communications to gather more simulation results from the WiFiCS Servlet

Access to advanced settings is held in a *Notecard* [281] which is part of the *inventory* of the All Seeing Orb. The *Notecard* is a text file held in an objects *inventory* and contains settings for the MAC protocol and sensing range of the nodes. The *Notecard*, its settings and comments are shown in Figure 80. Users can easily edit these advanced settings using the Second Life inbuilt editor. The comments provide alternative settings and the format they should be entered in.

```
dsr
11
75
//1st line is routing protocol, other options are: aodv, dsdv,flooding,tora
//2nd line is MAC protocol, other options are: 11, aloha,tdma - 11 is 802.11 protocol
//3rd line is the sensing range
```

Figure 80: Advanced settings available to the user in Second Life

6.7.4 Execution Scheduler

The execution scheduler triggers events at a certain time, as specified in the resulting network trace of a scenario, and controls the playback of a scenario. Its input is the compressed multi-dimensional text data that details each network event as shown in Figure 77 and in more detail in Figure 81. The structure of the execution scheduler is similar to previous schedulers written in ActionScript 2.0 and ActionScript 3.0.

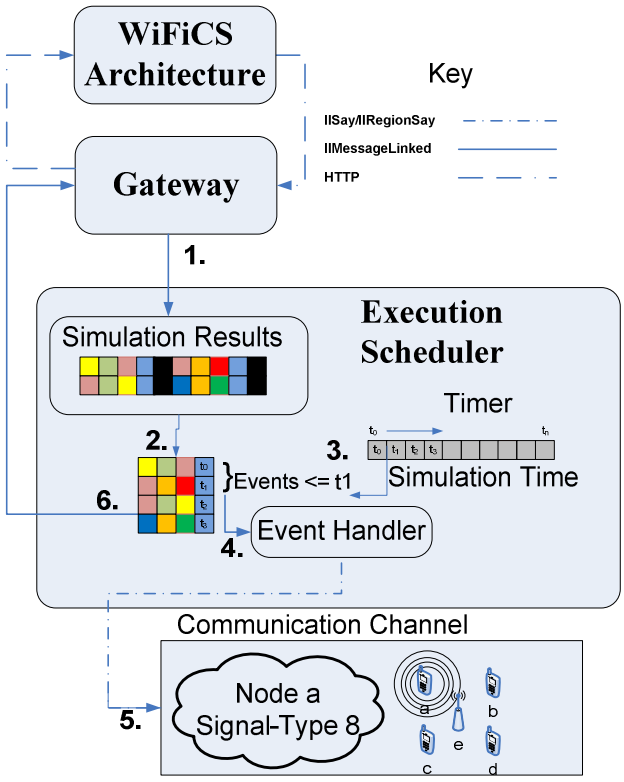


Figure 81: Operations of the execution scheduler developed in Second Life

Figure 81, step 1

The scheduler receives a string which is the compressed scenario trace providing information on every network event in the scenario (see section 6.7.3).

Figure 81, step 2

The scheduler then splits the string into a list of lists; as such, the compressed scenario trace is a multidimensional list. The split is performed by generating a list of colon delimited strings. These

strings are further split to provide a list that relates to an event in the simulation. The functions that perform these splits are shown in Figure 82.

```
//splits the new events into a list from the raw string from output = ls
list addNewEvents(string moreEvents){
   return llParseString2List(moreEvents,[":"],[""]);
}
list convertStrEventToList(string aevent){
      return llParseString2List(aevent,[","],[]);
}
```

Figure 82: A sample of LSL code used to split the compressed NAM results into a manageable format

Once the list returned by the method "*convertStrEventToList",* the scheduler can extract key bits of information from the protocol such as the time of an event and the source node of a message. These pieces of information are accessed in accordance with good coding practices as shown in Figure 83.

```
float getTimeFromEvent(list anEvent){
    return llList2Float(anEvent,timeIndex);
}
integer getSourceNodeFromEvent(list anEvent){
    return llList2Integer( anEvent,sourceNodeIndex);
}
```

Figure 83: A sample of LSL code used to index into an event using the constants specified in Figure 78

Figure 81, step 3

The scheduler has a constant tick rate which allows the subdivision of the simulation into periodic events representing a slice of simulation time. Using a steady tick rate and a monotonic counter, the counter can be converted to simulation time by dividing by a constant to calculate the simulation time (Figure 84 line 7).

```
1.  integer ticks = 0;
2.  float simulationTime = 0;
3.  integer divisor = 1800;
4.  list eventList = [];
5.  timer(){
6.  ticks++;
7.  simulationTime=(float)((float)ticks/(float)divisor);
8.  executeAnyLessThanThis(simulationTime);
9.  }
```

Figure 84: LSL example code from the Execution Scheduler component in WiFiSL

The buffer *eventList* contains events to be executed on the next tick of simulation time. While executing, the events list is scanned progressively, halting once a value is encountered larger than the next simulation tick time (*maxTime*). Events that are to be executed are then executed using the *Event Handler* method. The executed events are deleted from the events list

```
executeAnyLessThanThis(float maxTime){
for(integer ptr=0; ptr<llGetListLength(eventList); ptr++){
list anEvent = convertStrEventToList(llList2String(eventList, ptr));
//if we go over the max time, everything before it will be deleted
//As it should be already handled below
if(ptr!=0 && getTimeFromEvent(anEvent)>maxTime){
eventList = llDeleteSubList(eventList, 0, ptr-1);
return;
}
…
else{
handleEvent(anEvent);
}
}
```

Figure 85: Sample code from the Execution Scheduler showing the selection of events to execute for a given maximum simulation time

Figure 81, step 4

The *handleEvent* method (Figure 86, line 2) extracts from an event its type, source ID (from ns-2) and its destination ID (ns-2). This method constructs a protocol message that instructs a particular listening component to animate an event. The particular node is indicated through the use of the Second Life key. There will always be a valid source address, the translation occurs in line 7. Not all network events have a valid destination address when broadcasting a destination address of -1 is used. As such, line 8 checks that there is a valid address before attempting the translation from ns-2 ID to a Second Life key. Once the animation message has been constructed, it is sent over the island-wide animation chat communication channel. All nodes listen on the animation channel and if they detect their Second Life key, as the originating key then the node initiates the animation of that packet.

```
1.  //called for immediate display/action
2.  handleEvent(list aevent){
3.  //also needs a translation from node id to SL key
4.  integer pktType = getPacketTypeFromEvent(aevent);
5.  string msg="";
6.  if(pktType != MOV){
        a.  msg = (string)pktType + ":" +                llList2String(slID,
            getSourceNodeFromEvent(aevent))+":";
7.  //if the destination address is broadcast (i.e. -1)
8.  if(getDestinationNodeFromEvent(aevent)!=-1){
9.  msg += llList2String(slID,getDestnNodeFromEvent(aevent));
10. else msg = msg + "-1";
11. llRegionSay(animationChannel, msg);
```

Figure 86: Sample code from the Execution Scheduler showing how events are transmitted to listening networking components on the island

The design choice for using *llRegionSay* for instructing a node to animate an event rather than the scheduler animating the event is due to the restraints in Second Life. Initially, to avoid any latency on the island and the simulator, networking components were linked together and communicated using *llLinkedMessage*. The limitation was that components could not be more than 5 metres apart which would limit the range of scenario constructions available. Another option was to have the Execution Scheduler *rez* the signal at the networking component's position. There are limits on the distance at which one object can *rez* another object; currently set to a 10 metre radius, which again restricts the range of simulation scenarios available and does not make use of the immersive environment of the

whole island. As such, by using the *llRegionSay* communication method and allowing each node to render the animation, island-wide simulations are possible.

Figure 81, step 5

The animation message is received by all the nodes on the island listening on the pre-calculated channel. Each node will receive the message and check if their Second Life key is a designated transmitter. If so, then the node will check the destination key. If a negative number is specified, the node will animate the signal in a broadcast manner. If a Second Life key is present, using a Second Life API call it will target a specific networking component using its x, y and z coordinates. The type of signal varies and a colour coding system is used to facilitate the recognition of signals, thus the signal type is checked before animation. More detail on the animation of broadcast and directional messaging can be found in section 6.5.

Due to the limitations imposed in LSL, the Execution Scheduler cannot maintain a full listing of the results of a simulation despite the large compression provided. Due to this space restriction, the scheduler will send a message requesting more simulation results. The extra results, when returned, are appended to the current list of results and this process continues until no more results are available. The user is capable of altering the pace of the animation by using the "Animation Board" that is available within the avatar's inventory alongside the All Seeing Orb and the Node.

6.8 *Evaluation and Methodology*

Evaluation of the system should not only come at the end of a system's creation. Evaluation should be carried out to help guide and correct the interface's design. This section discusses the wide range of methods for evaluating a user's experience of a system. The important factors in evaluation have been divided into three sections:

1. For user evaluation to be performed, a sample of users must be chosen. It is important that the users are representative of the intended target group of the system.
2. The users should have provided informed consent and be aware of their rights throughout. A user evaluation session should be a positive experience for all involved. The user has a right to know what information is being collected and how it will be used. Privacy is an important issue and names should not be recorded together with viewpoints expressed. At any point during the evaluation the user can terminate the session without having to give a reason and without fear of recrimination.
3. Another important consideration when evaluating software is which version of the software to present to the user. It is important to evaluate the software early in the development lifecycle. Often if the developer is the only user evaluating the system, certain tasks become too easy to achieve with their background knowledge. An outside user is required to review the system. If the evaluation is left until late in the software lifecycle then it will be more difficult to improve the software. Further, the later in the software lifecycle a development flaw is left,

the more expensive in time and money it will cost to fix [282]. Early evaluation by outside users is an important technique for recognising difficult tasks and problem points in an interface. This early evaluation allows the lifecycle to be guided towards the end users' needs.

The evaluation session can be designed in several ways. The least rigid is an open-ended session where a user is not provided with any information or goals to achieve and is allowed to experiment with the system. A more structured approach is to provide the user with a set of tasks to achieve. Selecting appropriate tasks for a user to achieve involves the right amount of required background knowledge and a task that is not specially adapted for the designed system. The tasks should be ordered from simple to hard. This allows the user the opportunity to master the system before attempting the more difficult tasks.

Evaluation can be performed through the observation of a user interacting with the system. The person evaluating the system records the tasks attempted by the user and the methods used to achieve such tasks. In [283] a discussion on which data to collect is provided. The data gathered by observation is differentiated into process and bottom data. The process data is the steps the users are taking to solve one of the tasks set. This process data is a step-by-step report on the activity of the user. The bottom data is the measurement of how many errors were made, which tasks were achieved and the time taken to do so. It is important to collect both types of data, and in particular, to focus on the process data in early testing. If the process data is ignored and only bottom data is collected, there is no way to discern where users went wrong.

The thinking-aloud method [284] of evaluation involves the user of the system verbally detailing each step as they attempt to complete a task. The user should be fully aware that it is the system and not their mental processes that is being evaluated. The role of the observer can be active or passive. The observer can prompt the user with questions such as "tell me what you are thinking" or "keep talking". These are open-ended question and allow the user to continue giving feedback. The observer should not attempt to guide the user to a feature they may have missed on the interface.

A passive evaluation can be performed through the examination of records maintained by the system that the user has interacted with. Recordings of the actions on the system can be replayed and analysed for achievements and mastery of the system. Frequency and duration of use can be calculated and compared between multiple users.

A more active form of evaluation involves using the System Usability Score (SUS) [257]. SUS gives a number between 1 and 100 that is derived from a short questionnaire. It can be used to provide a score that is comparable to other users and system versions. The questionnaire alternates between positive and negative questions concerning the use of the system and how strongly or negatively the user agrees with the statement. This system was devised to enable users, often tired after a lengthy evaluation session, to provide feedback in a compact way.

A range of debriefing interview techniques can be used to evaluate a user's experience of a system. There is anecdotal evidence that suggest that users will remember the solution to a problem rather than the problem itself. This can lead to a user struggling with a difficult system and not reporting this to the interviewer. This is an example of why interview results alone should not be relied upon. Interviews are a useful technique for getting detailed feedback and fit well as a debriefing technique.

The most rigid approach is a structured interview. In a structured interview, there is a fixed order of questions given to the user. The user may respond either from a given set of responses or a more open-ended approach where opinions can be articulated. Due to the ordering of questions being maintained, the possibility of influencing answers through juxtaposition of questions is ruled out.

A semi-structured interview provides a more flexible approach to the interview process [285]. The interviewer may have several aspects of the software they wish to gain feedback on but there are no fixed set of questions. The lack of a fixed set of questions allows a more natural discussion to flow and can be steered towards areas of particular interest of the interviewer. An unstructured interview allows the adaptation of a set of questions based on the user's capabilities. The user is also not restricted to a fixed set of responses, unlike in a structured interview [286].

6.9 *WiFiSL Evaluation*

Based on the approaches to evaluation detailed in section 6.8, this section provides the methods used for the evaluation of WiFiSL. The evaluation is three-fold as it provides information before the user interacts with the system, records their interactions and achievements during the session and ends with a debriefing section. Due to the novel nature of a MUVE, the number of users has been kept low. Using a lower amount of people allows greater focus on each one, which is important in the development of interfaces in such a novel environment. An objective of this evaluation session is to ascertain the applicability of using a virtual world for exploratory learning. The second objective of this evaluation session is to observe and measure the design of a 3-dimensional interface in use.

A series of tasks has been set for users of the WiFiSL. A task-based approach reflects the purpose of the WiFiSL in that it must enable users to complete worksheets on a protocol and demonstrate understanding. The tasks to complete are:

- to deploy networking components into the environment
- to construct a scenario involving two nodes transferring a file
- to alter the routing protocol
- to alter the MAC protocol and describe any changes
- to modify the sensing range and observe the difference
- to modify the RTS/CTS threshold and describe, with the use of supplied statistics, any changes observed

These tasks were given to the user (see appendix 1.7) and an observer recorded the steps the user took to achieve the goals. This data is known as process data. This process data is useful for future refinements in the interface of the WiFiSL. A measurement is also recorded of the number of tasks achieved; this is the bottom-data. The time taken to achieve the tasks is not recorded as the environment is an exploratory one and the various tasks may not be attempted in a coherent order.

Whilst the user is attempting to complete the tasks, the thinking-aloud protocol is used to monitor which parts of the system are causing problems for users. No audio recording is made of this protocol as it may intimidate the user. The thoughts provided by the user are recorded by the observer on paper. Each user has the opportunity to see all data collected at the end of the evaluation session and can remove their permission for its use or reproduction.

After the user has finished using the system, a System Usability Scale (SUS) questionnaire is provided. This can be compared to values gathered at previous laboratory sessions for WiFiVL. A similar educational questionnaire is used to establish the students' level of interest and perceived educational value about the system.

Finally, an unstructured interview is used to discuss the user's feelings about the system. An unstructured approach is more suited here as the questionnaire will already provide a structured approach. An unstructured interview will have several topics for general discussion: views on virtual worlds, difficulty in using WiFiSL and suggested improvements.

6.9.1 First Evaluation Session

The first evaluation session was performed with the participation of a post-graduate student in the School of Computer Science at the University of St Andrews researching networks. The student was under direct supervision and asked to perform various tasks *in-world* using the virtual laboratory. Before attempting these tasks, an introductory talk was provided detailing the uses of the components and ways to construct a scenario.

Initial problems involved the complicated rights management in Second Life. The default behaviour of Second Life is to restrict copying an item that has been set for sale. When the node and orb were set for sale, both had to be explicitly marked as available for copying by the student.

The student was quickly able to join nodes together. The notion of scaling traffic time to real time was seen as confusing.

This was the first use of WiFiSL and despite problems, the user was able to deploy networking components, construct a scenario and review the resulting animation. The version of WiFiSL used was one where all actions and durations in the virtual world were scaled from the user-perceptible time frame of minutes to a simulator-suitable timeframe of milliseconds. The first set of results from this session is the process-data which provides us with information on the user's processes. The process-

data presented here highlights problems with this version of WiFiSL which were addressed before continuing evaluation with other users.

6.9.1.1 Process Data

- Multiple permission problems when transferring the networking components into the user's *inventory*. The networking component in their inventory was unable to be copied more than once.
- Lack of textual support. The user was unable to review previous signals transmitted on the network in the chat history box.
- The user expressed difficulty in relating to the simulation time. This was due to the lack of simulation time being displayed when the signal is animated. WiFiVL I and II had shown the time at which a signal was transmitted in a text console.
- Difficulties in traffic construction expressed and a misunderstanding of how to express duration in a link.
- The console interface was used and became a powerful way of expressing more complicated scenarios. To construct a scenario in the console requires the sender and receiver identifier. There was no way to find a networking component's identification number in the virtual world.
- User expressed confusion during long gaps in the playback. The user was unsure if this was due to gaps in the simulation or if all the results had been shown and playback was finished.

Task Name	Status at end of session
Deploy networking components into the environment	Successful
Construction of a scenario involving two nodes transferring a file	Successful
Alter the routing protocol	Unsuccessful
Alter the MAC protocol and describe the changes observed	Unsuccessful
Modify the sensing range and observe the difference in	Unsuccessful
Modify the RTS/CTS threshold and describe, with usage of supplied statistics, any changes observed	Unsuccessful

Table 21: Task results – bottom data for first evaluation session

Question	Strongly disagree				Strongly agree
I think I would like to use this system frequently				X	
	1	2	3	4	5
I found the system unnecessarily complex				X	
	1	2	3	4	5
I thought the system was easy to use			X		
	1	2	3	4	5
I think that I would need the support of a technical person to be able to use this system			X		
	1	2	3	4	5
I found the various functions in this system were well integrated				X	
	1	2	3	4	5
I thought there was too much inconsistency in this system			X		
	1	2	3	4	5
I would imagine that most people would learn to use this system very quickly			X		
	1	2	3	4	5
I found the system very cumbersome to use			X		
	1	2	3	4	5
I felt very confident using the system			X		
	1	2	3	4	5
I needed to learn a lot of things before I could get going with this system			X		
	1	2	3	4	5

Table 22: SUS results from the first evaluation session, the score for this response is 52.5%

This SUS score is 52.5/100 and reflects the difficulties the user had in interacting with the WiFiSL. The perceived educational value was 66%.

6.9.1.2 Unstructured Interview

The unstructured interview was used to reflect on the difficulties encountered, suggested improvements and areas of the interface that were enjoyable to use. The user expressed a real sense of engagement and immersion into the virtual world. They found the interface for placing networking components *in-world* and creating connections to be "easy to use". Another aspect that was discussed was the advantage and interest in observing the signal interactions and the ability to enjoy different angles during playback. A focus point for the interview was the difficulty in understanding the translation of in-world time to simulation time.

6.9.1.3 Analysis

This version of WiFiSL scaled the length of user interaction in the virtual world to simulation time. As an example, if a user were to leave two networking components connected for 10 seconds before closing the connection, this would translate to 0.5 seconds of traffic communication in the simulation. Another example of the scaling was that the time between two connections was also scaled. So if a user were to wait 2 minutes between closing the first connection and creating a second one, then the

time between the two connections in the simulator would be scaled to 2 seconds. These notions confused the user and created difficulties in understanding the scenario they were creating.

The notion of translating *in-world* time to simulator time was confusing and so has been removed from WiFiSL. It was as a result of this session that users now construct connections between nodes and a fixed file size is transferred between nodes. The times between transfers are scheduled by the All Seeing Orb to occur 0.002 seconds after each other. This evaluation session has shown that the bottom data would not have shown up the problems that the user was encountering. It is therefore important to maintain both process and bottom data.

6.9.2 Second Evaluation Session

The second evaluation session took place after the WiFiSL had been altered to incorporate suggestions and correct performance issues highlighted in the first session. The user was a post-graduate student in the School of Computer Science at the University of St Andrews researching networks.

6.9.2.1 Process Data

- The user appeared comfortable in creating larger scenarios.
- Traffic connections between networking components were made quickly using the visual interface.
- The console was used frequently to construct traffic links and review the current state of the scenario under construction

Task Name	Status at end of session
Deploy networking components into the environment	Successful
Construction of a scenario involving two nodes transferring a file	Successful
Alter the routing protocol	Successful
Alter the MAC protocol and describe the changes observed	Successful
Modify the sensing range and observe the difference in	Successful
Modify the RTS/CTS threshold and describe, with usage of supplied statistics, any changes observed	Successful

Table 23: Task results - bottom data from the second evaluation session

Question	Strongly disagree				Strongly agree
I think I would like to use this system frequently					X
	1	2	3	4	5
I found the system unnecessarily complex		X			
	1	2	3	4	5
I thought the system was easy to use					X
	1	2	3	4	5
I think that I would need the support of a technical person to be able to use this system	X				
	1	2	3	4	5
I found the various functions in this system were well integrated				X	
	1	2	3	4	5
I thought there was too much inconsistency in this system	X				
	1	2	3	4	5
I would imagine that most people would learn to use this system very quickly					X
	1	2	3	4	5
I found the system very cumbersome to use	X				
	1	2	3	4	5
I felt very confident using the system					X
	1	2	3	4	5
I needed to learn a lot of things before I could get going with this system	X				
	1	2	3	4	5

Table 24: SUS score for the second evaluation session is 95%

The SUS result report by this user was 95/100. The educational consideration was also high at 92/100. Both of these scores are a large improvement on the first evaluation session performed on an earlier version of the software.

6.9.2.2 Unstructured Interview

The user reported that there was a great advantage in using Second Life due to the amount of residents who are already familiar with a virtual environment. There was generally positive feedback concerning the user interface with the user reiterating it was easy to set up and use.

6.9.2.3 Analysis

The student was comfortable with both the 3-dimensional and console interface, generating simulations quickly. The simulations grew to a larger scale on the island as well as being able to review the statistical analysis provided by the WiFiCS. The user gave an SUS score of 95 and an educational consideration score of 92. This is a clear improvement on the previous version of the WiFiSL.

6.9.3 Third Evaluation Session

No significant changes were made to the system between the second and third evaluation session. The user was a post-graduate student in the school of Computer Science at the University of St Andrews researching networks.

6.9.3.1 Process Data

- User reported confusion about the number of debugging messages supplied to them by the networking component.
- The colour of some signals was not shown on a key. The information included in the console allowed the user to calculate which signal type corresponded to a certain colour.
- Easily added networking components into the virtual world

Task Name	Status at end of session
Deploy networking components into the environment	Successful
Construction of a scenario involving two nodes transferring a file	Successful
Alter the routing protocol	Successful
Alter the MAC protocol and describe the changes observed	Successful
Modify the sensing range and observe the difference in	Successful
Modify the RTS/CTS threshold and describe, with usage of supplied statistics, any changes observed	Successful

Table 25: Task results - bottom data from the third evaluation session

Question	Strongly disagree				Strongly agree
I think I would like to use this system frequently				X	
	1	2	3	4	5
I found the system unnecessarily complex	X				
	1	2	3	4	5
I thought the system was easy to use				X	
	1	2	3	4	5
I think that I would need the support of a technical person to be able to use this system		X			
	1	2	3	4	5
I found the various functions in this system were well integrated					X
	1	2	3	4	5
I thought there was too much inconsistency in this system		X			
	1	2	3	4	5
I would imagine that most people would learn to use this system very quickly					X
	1	2	3	4	5
I found the system very cumbersome to use	X				
	1	2	3	4	5
I felt very confident using the system					X
	1	2	3	4	5
I needed to learn a lot of things before I could get going with this system		X			
	1	2	3	4	5

Table 26: SUS result from the third evaluation session, the score was 87.5%

6.9.3.2 Analysis

During development there were debugging messages inserted into the source for the networking component and the all seeing orb. These messages only communicate with the owner of the item but were sending too much internal state information. Most of these owner debugging messages have been removed. The SUS score was 87.5% and the Perceived Educational Value was 76%.

The results from these evaluation sessions is summarised in Table 27.

WiFiSL	
Number of Participants	3
Evaluation Methodology	Process Data, System Usability Scale, Perceived Educational Value and Unstructured Interview
System Usability Scale Average	78.33%
Perceived Educational Value Average	78%

Table 27 Summary of the evaluation methodology and results obtained during the WiFiSL evaluation

6.10 Implementation Issues and Limitations

While Second Life provides new, engaging and exciting forms of interactions with users, there are limitations on its abilities and that of ns-2. This section details the limitations of the designed WiFiSL system and where possible a solution is proposed for use in future work.

There is no import ability or simplified Remote Procedure Calls (RPC) for interaction between scripts. The transfer of information is achieved through chat channels and message parsing. There are two inter-script communication techniques:

- **Link Message** – A Link message is sent to all scripts inside the same prim. When multiple prims are joined together, the information sent is received by all scripts in all the connected prims that have set a listen handler for linked messages. Link messages can include a sender number and the Second Life key of the originating prim. The definition is:

  ```
  link_message(integer sender_num, integer num, string str, key id);
  ```

- **Say** – A message can be transmitted on any of the 4,294,967,294 communication channels and can be broadcast to entire regions or local areas. In order to hear a message, the script must have registered a listening handle on that channel. The information passed contains a string of information and the originating Second Life ID. Various objects can be cast as strings, then cast back to their originating type. This can lead to various problems. When defining a listen handler, filters can be applied such as receiving from only a specified Second Life ID. There are performance penalties for using the command Say. These penalties induce latency in the island's simulator. Hence, the preferred method for inter-script communication within an object is to use a Link Message as described above. The definition of the listen method is as

follows:

```
llListen(integer channel, string name, key id, string msg);
```

The choice between these communication methods is a critical one because components in WiFiSL have to display network events at accurate times and penalties such as simulator lag and enforced time penalties can severely affect performance.

Minerva Island is the experimental test bed of the system and like real life wireless networks, there are obstructions such as buildings and the topography. The release of ns-2.33 provided no additional ability to model a network scenario where a physical obstruction prevented nodes from communicating. For example, as illustrated in Figure 87, node A should not be able to communicate with node B due to a physical barrier.

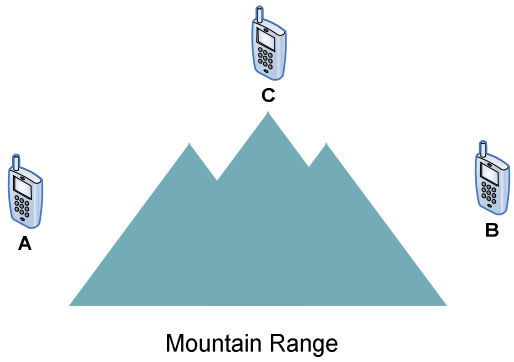


Figure 87: Node A and B can only communicate with node C and not with each other due to an obstruction

To resolve this problem, an alternative simulator can be used that allows different receiver strengths to be specified between nodes. In the example of Figure 87, the signal strength between A and B can be set to zero whilst A to C and C to B can have a value high enough for communication. Currently, individual node defined signal strength is not possible under ns-2.33 as the signal strength is set for all nodes in a simulation.

Although an alternative simulator can be used, calculating obstructions in Second Life will still have to be performed. Second Life lacks a topology API to query the presence of a line of sight between two objects. Objects in Second Life can detect collisions and create projectiles aimed at another object's location. In Figure 88, node A creates a physical projectile and sends it towards node B (step 1). Node B detects a collision and can inform Node A through a region chat channel that it has received its projectile and therefore the two objects have line of sight (step 2).

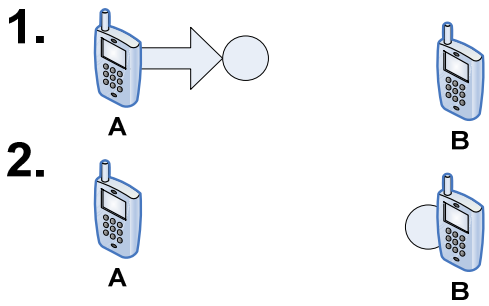


Figure 88: An object creating a projectile and targeting another object followed by a collision

In Figure 89, Node A again transmits a particle with the destination as Node B's x, y and z location. Due to an obstruction in step 2, the projectile never collides with Node B and therefore there is no line of sight.

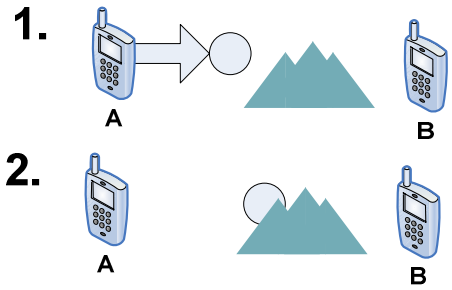


Figure 89: An object creating a projectile and targeting another object with the collision occurring at an obstruction indicating no line of sight between the two nodes

Second Life imposes restrictions on *rez*zing objects larger than 10 x 10 x 10 metres, informally referred to as Mega Prims [287]. This created design issues in the system where a 75 metres range wireless scenario cannot create the equivalent 75 metres expanding spherical object. While they are not prohibited, their status is in doubt with future support unknown. Mega Prims are not illegal to own on private estates. Certain Second Life clients restrict the ability to see or use a Mega Prim. The reason for disabling such a useful feature is due to its facilitation of nuisance behaviour. Discussions are ongoing as to the future of official support for Mega Prims. Due to the lack of official support at the time of the system design and implementation, they were not used in the system.

Shared land ownership on an island means owners can prevent unwanted dropping of items. Whilst WiFiSL currently supports the use of a simulation across the island, land ownership may create some issues on other islands. As such, it would be impossible to drop a networking component on a more restrictive land policy. This has caused some difficulties even on a private land such as Minerva Island.

6.10.1 **Draw Distances**

A limitation in many virtual environments is the restriction on draw distances. The draw distance is a setting that specifies the maximum distance from the *avatar*s that an object will be drawn on the screen. In WiFiSL, where a scenario may be taking place over the entire island, a small draw distance will restrict the viewing of the resulting simulation. One area that limits the setting is the capability of the graphics card. With a large draw distance many objects may be required to be rendered by the card and this puts strain on the graphics card. In a MUVE such as Second Life, where much of the content is user derived and thus cannot be pre-downloaded or distributed via alternative methods, a large draw distance will create intensive use of the network. The large draw distance will require a signficant amount of information about all the objects in the area and creates strain both on Linden Labs and the end-user bandwidth requirements. The client, Second Life Viewer, may also limit the draw distance in an attempt to lower the bandwidth and hardware intensive requirements; as such, this is a setting that requires overriding. This is a disadvantage to a MUVE whereas in a 3-dimensional environment such as that in World of Warcraft or Counter Strike: Source, where the content is static, the draw distance is very large and as such can create a more immersive environment. As network speeds continue to increase draw distance restrictions will decrease. A significant amount of traffic is inevitably generated when large groups of people congregate in Second Life [288]. Other alternatives may be the creation of a proxy server capable of distributing Second Life assets to those in its network. Therefore, a classroom full of users would only create a single request from Linden Labs for an asset's information. Reflections on Second Life suitability

Scripting in a MUVE must strike a balance between allowing complex behaviour and yet be easy enough to grasp for quick adoption by average users. While this allows many people to quickly create and share scripts, it also prohibits or restricts more advanced usages, such as this WiFiSL. Whilst Open Simulator has similar restrictive elements, the underlying source code for the island and simulator can be altered and recompiled, unlike the closed-source nature of Second Life.

The use of a MUVE may be viewed as adding another barrier to learning as it may take some time to become proficient in its use. Students are, however, quick adopters of new technologies and many are already active in a MUVE. A 3-dimensional interface may also be viewed as more intuitive than using 2 dimensions as it is analogous with our daily lives and is an environment to which we are most accustomed.

Second Life has a strong economy requiring strict rights management over user-generated content. Other aspects of Second Life are also limited, for example the uploading of images to use as textures are limited through pricing. This pricing is used to cover the cost of storage by Linden Labs. Due to pricing, uploads, whilst not severely restricted, are not widespread. Projects and buildings in Second Life are designed to keep costs low and there is extensive reuse of textures and objects. No programmatic automatic upload based on user preferences is currently feasible. This is an area where

Open Simulator provides a viable alternative as uploads can be made free through hosting the simulator on local storage and processing hardware.

Without the ability for efficient code reuse and standard ways for inter-script and inter-process invocation, large scale projects in virtual worlds will continue to be difficult. There is no current widespread adoption of library scripts that perform many basic tasks. The availability of such libraries in the open source world such as Hibernate and JSF has enabled developers to quickly design more complex programs that can abstract away lower level implementation issues such as transaction management of a database.

Large scale projects also quickly develop hard to manage bugs due to run-time only checking of the script. This is not facilitated by the required use of the provided scripting environments in the Second Life Viewer. As developers move into Second Life, other alternative scripting environments have been created [289]. A capable programming environment is the lsleditor [290]. This provides a Visual Studio approach of creating a solution for a project that will group together scripts, animations and *Notecard* elements. Scripts are checked for errors and can be run inside the editor. Interactions are provided such as a console for inputting text on a certain channel that the script will be monitoring. *Avatar* interactions are also provided with the ability to debug the script. Another key feature is the ability to interact with a CVS [236]. This is an important step in allowing more widespread development of code for a MUVE. The editors are limited in their ability to display animations to the user and as such, for a very visual project such as WiFiSL, the editors were of limited use. An editor that would provide a physical view of the objects and the animations it would produce would be an impressive, though difficult interaction to achieve.

For use in a classroom, it may be useful to have an entire island dedicated to each student. In Second Life, however, this would require the purchasing of several islands at great expense. Each island can only be modified *in-world* and as such each one would have to be individually configured before use. The use of Open Simulator allows multiple instances of an island to be created at no additional cost. Each island can be specifically designed for a certain scenario, e.g. one island where the hidden node is already configured. Using a central control computer, an instance of the Open Simulator can be created and students given the address to connect to.

Second Life's economic model will ensure future interest and activity from those hoping for financial gain through user-generated content. Such economic reward can often provide an increased incentive for more users to remain connected to Second Life and attract new users into the MUVE.

6.11 *Transposition of WiFiSL to WiFiOS*

As the WiFiSL project progressed, limitations of Second Life arose. The cost of renting an island is high and the cost for uploading new material, whilst not entirely prohibitive, is restrictive. Due to the lack of charge for uploading material in Open Simulator, a large amount of slides and other types of objects can be created and uploaded. Open Simulator had also progressed during our development time

in Second Life. Open Simulator, whilst still in an alpha, had become more stable with an almost complete implementation of Linden Scripting Language (LSL).

Ownership of Minerva Island in Second Life was shared amongst several users. The land was subdivided with WiFiSL occupying a small portion of land. During evaluation, it was clear that hosting more than one or two users at a time would be very difficult due to the small space. Open Simulator allows for expansion to a whole island. This allows for more students to use islands simultaneously and to give more space for other features.

Open Simulator supported a range of new commands that are not possible in Second Life such as the use of web texturing and the ability to draw shapes on primitives. There is also the possibility of designing your own commands due to the open source nature of Open Simulator.

Open Simulator, as of version 0.5.9, has the ability to export a whole island. Every primitive on the island, their inventory and terrain data can be exported as a single file. The single file is called an OpenSimulator Archive (OAR) [291]. An OAR file uses XML for information storage. The OAR file can be shared between any Open Simulator installations allowing duplication of islands.

As a result of all these considerations, the virtual laboratory was transposed from Second Life to Open Simulator. Due to the strict use of LSL when developing the virtual laboratory in Second Life, the transposition from Second Life to Open Simulator was a straightforward one. Objects structure and textures were exported from Second Life using the Meerkat viewer [292]. Scripts were copied and pasted from the SL viewer into primitives in Open Simulator.

6.12 *WiFiOS Design*

The design of WiFiOS was such that it could be easily shared using an OAR file. This required that no modifications were made to the underlying Open Simulator source code. Since there is no restriction on the number of islands available, a whole island was dedicated to WiFi. The island, named Aeolus, hosts the WiFiVL and a large range of supporting materials.

The island was designed with a centre circle with signposts that direct to different parts of the island. The signposts use the *osTeleport* [293] command to transport *avatars* immediately to different parts of the island. A more open approach such as open exploration is also encouraged as Avatars are free to fly around the island. An example screenshot of the island (Figure 90) shows the centre building with links that connect the different areas.

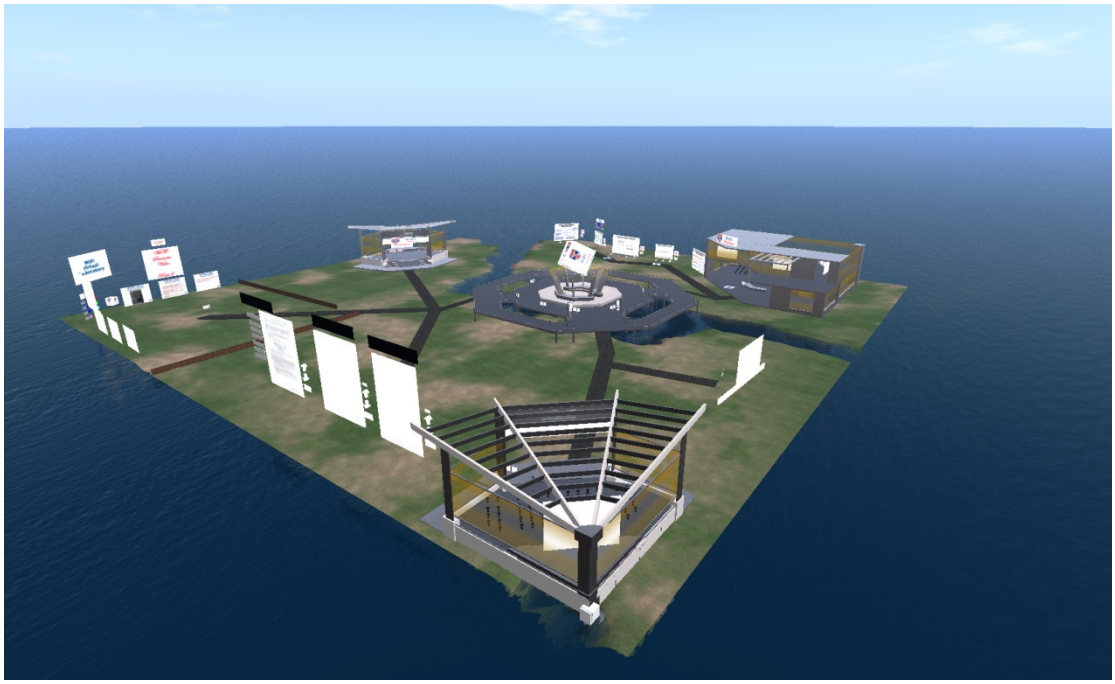


Figure 90: Screenshot of the WiFiOS island Aeolus

With an entire island available to WiFi, supplementary materials were designed, similar in nature to the support materials available on the WiFiVL I host site. The support materials constructed ranged from traditional lecture slides to more interactive frame format displays.

Several of the building designs are taken from the OpenVCE [294]. Only their structure and not content has been used in this dissertation. The OpenVCE buildings have retained their banner where applicable. The materials were made available as the OpenVCE had distributed an island using an OAR file.

A lecture hall was created with a range of lectures available, a screenshot of which is shown in Figure 91. The lecture theatre has components to move slides forward, back and reset the lecture. Underneath the podium are a range of primitives that represent different lectures from a range of sources [295-296]. By clicking on one of these primitives, a new lecture is loaded onto the main display. The lecture theatre code is written entirely in LSL and can easily support new lecture slides. Slides can be created from Microsoft PowerPoint by choosing "Save as JPEG".

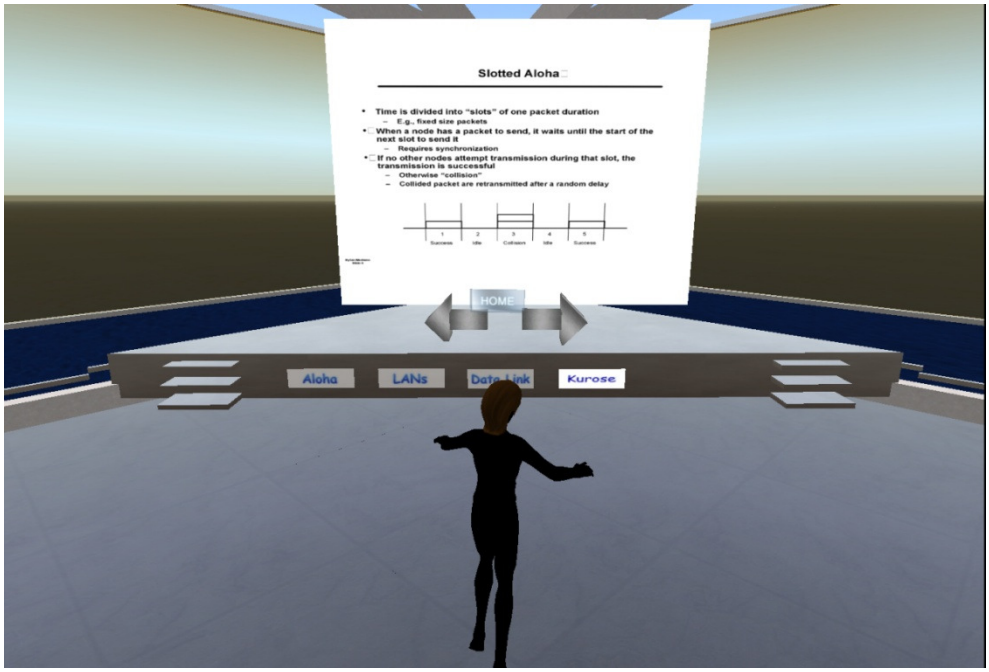


Figure 91: Screenshot of the lecture theatre with a range of lectures available for selection by the *avatar*

A WiFi museum was designed which displayed the history of wireless communications. The museum is split into time periods, from 1887 – 1915 to 1916 – present. A timeline is presented in three parts, with display boards showing information on important people or inventions. The museum starts with Heinrich Hertz and the advancements of Nikola Tesla and finishes with the publication of the IEEE 802.11 standard and WiMax.

An area of the island was also dedicated to the novel display of influential documents. The first is a patent awarded to a famous actress, Heddy Lemarr, for a Secret Communication System [297]. The second document is the influential paper by Shannon on a Mathematical Theory of Communication [298]. The third document is the IEEE 802.11 2007 standard [158] with bookmarks to relevant parts such as the frame structure, RTS/CTS structure and mechanism. All the pages could be viewed from within the island and links were also provided for viewing in external PDF readers. The documents each had a page up and down button as well as a home button.

An interactive frame format display was developed; a screenshot is shown in Figure 92. The IEEE 802.11 frame was subdivided into parts. Each part had a size representative of the number of bits used. Where a part of the frame is subdivided (e.g. Frame Control with sub sections of Protocol and Type etc.) the components are made visible to the user. When the user clicks on a part of the frame, a display board provides information on that frame section.

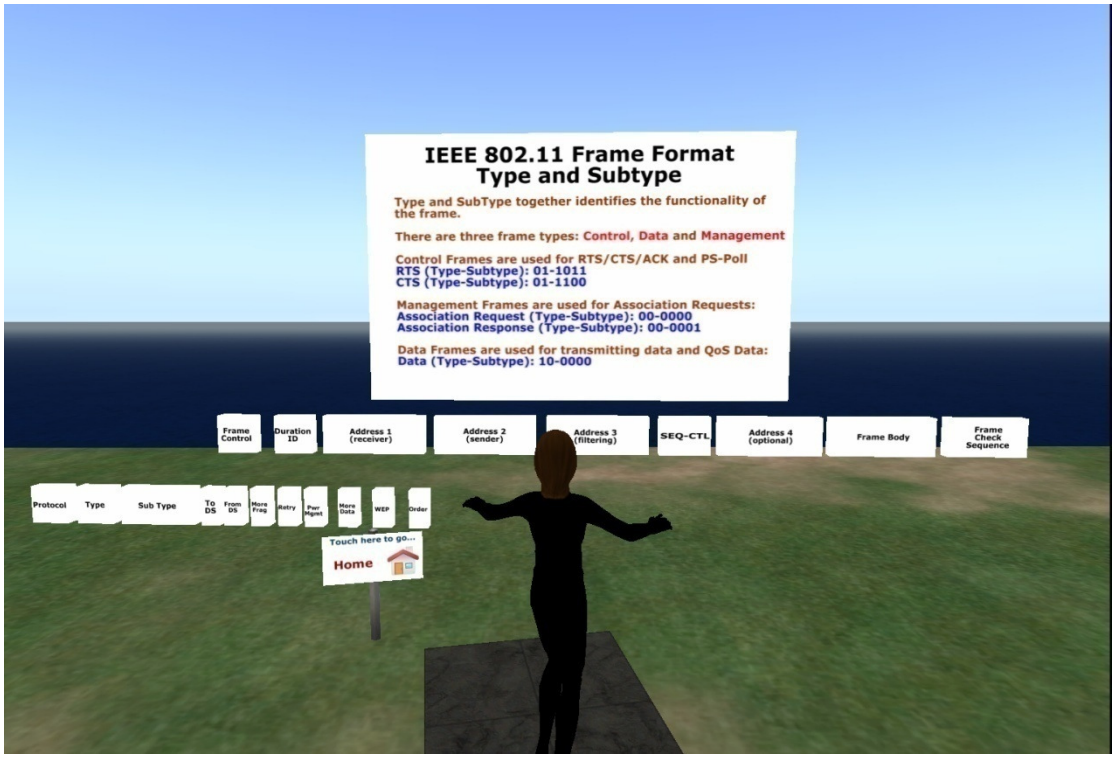


Figure 92: Screenshot of the interactive IEEE 802.11 frame display on the island Aeolus

A rich mixture of media was used on Aeolus, an example of this is the inclusion of video lectures. To enable videos, a suitable storage place had to be used. To facilitate this FreeBSD 8.0-RELEASE was installed on a separate server named minervamedia. The media server uses Darwin Streaming Server [299]which is an open source RTP/RTSP streaming server [300]. Video lectures have been incorporated into Aeolus through the WiFi cinema section. The video lectures are taken from MIT Open Courseware [301]. To play a video, the land must be partitioned as only one video per parcel is possible. As such, each video display is stored on its own parcel. The parcel's media URL is set to the URL for the video stored on minervamedia. Once the user clicks on the display, the video is shown. There are two other videos currently on Aeolus that demonstrate how to use the WiFiVL.

Another multimedia approach to the education of WiFi networks is the use of stock animations and display boards. These boards, as shown in Figure 93, are used to animate and describe features of wireless networks. The students can walk through the introduction of Carrier Sensing Multiple Access, the Hidden and Exposed Node Problems and their ultimate solution in RTS/CTS with data exchange. The animations are rendered using the same equipment as the WiFiVL uses and helps to build familiarisation with the colour coding of particular packet types.

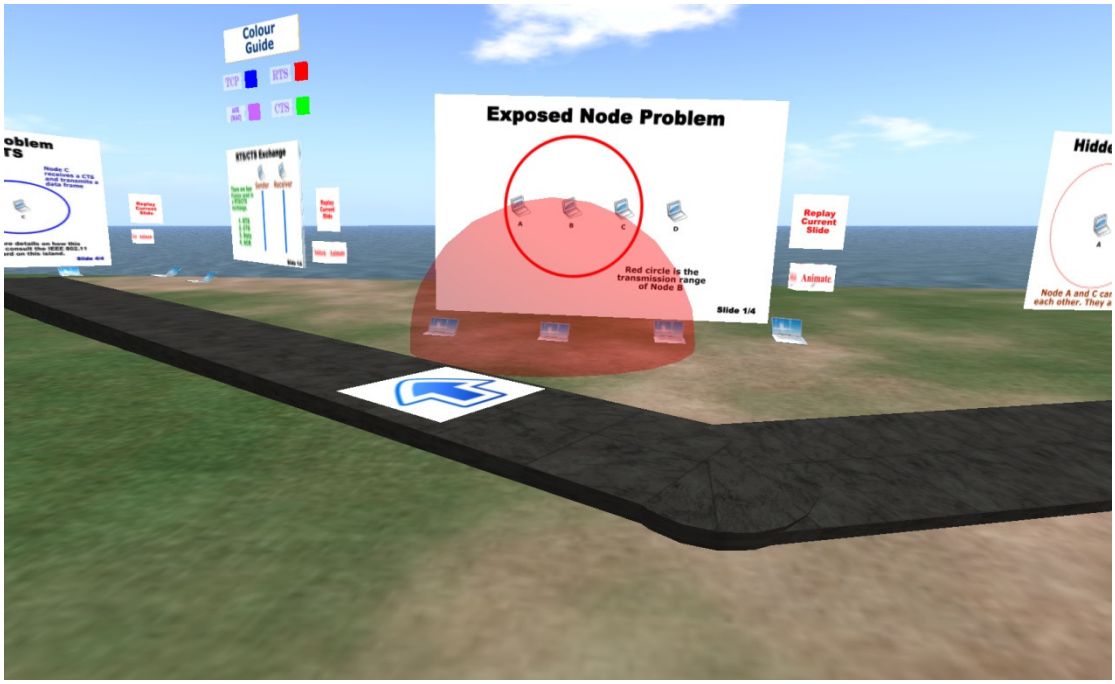


Figure 93: Example demonstration of the interactive education display for the Exposed Node Problem

6.13 *WiFiOS Evaluation Session 1*

An evaluation session was designed to guide students through each of the areas available on Aeolus. The students were Undergraduates at the School of Computer Science in the University of St Andrews who responded to a lab announcement for volunteers. A worksheet was provided (see Appendix 1.9). In this worksheet the students start with familiarisation of virtual worlds and the movement within such an environment. Next, students are guided towards the lecture theatre and asked to find some information on CSMA and the Aloha protocol.

After this, the worksheet advises students to review the problems and solutions section of Aeolus. This section provides the animation of aspects of CSMA and the hidden and exposed node problems. Once students have worked along this section, there is a question about frame formats. The frame format section of the island provides an interactive frame structure and display board containing information. The students gather information on the different fields here and answer questions such as which components of the frame provide support for fragmentation. Once these sections have been covered, the students can construct their own experiments using WiFiVL. This enables the display of the mechanism for transferring data, the ability to modify the MAC protocol and toggling the RTS/CTS frames.

The students were a mixture of 3rd and 4th year honour students studying for Internet Computer Science degree at the University of St Andrews. The students were not able to use any external resource such as a web search engine to provide answers to the question sheet.

6.13.1 Observer Notes

The session took place in February 2010 and lasted two hours, with five students using the same computer configuration of CentOS release 5.3 (Final) 2.6.18-128.7.1 x86_64 Intel Core 2 Quad CPU Q6600 @2.4GHz. The client viewers were the Second Life viewer 1.23.5 (136274) October 14 2009 13:31:00 Release Candidate. The server that hosted an instance of the WiFi Island was a Windows 7 Professional, Intel Xenon CPU E5504 @2.00GHz (2 processors) 16GB RAM, 64bit CPU and OS. The island was loaded using a single shareable OAR file.

The default configuration of Second Life viewer was modified to increase the maximum resolution and draw distance. The default draw distance limited the navigation of users around Aeolus as they were unable to see structures further than 50 metres away. It appeared as though some slides were slow to load but did not hinder progress by a great degree.

Aeolus had all permission restrictions removed. This allowed any student to modify or create any object on the island. While some slides and video displays were deleted, this was mostly done by accident and easily remedied. The students enjoyed having freedom over their progress through the worksheet and their environment. Although not specifically mentioned in the worksheet, a great interest by the students was shown in video content as well as the WiFi museum.

Three of the group organised themselves whilst reviewing the slides so that they only moved on when everybody was ready. The organisation was done through a mixture of chatting in the laboratory and *in-world* messages.

An Open Simulator bug [302] that would not fully delete objects from the environment was observed during this session. The issue can be fixed by users reconnecting though the students did not find it to be a large impediment. This was fixed by updating to the latest release of Open Simulator.

One student, new to the study of wireless networks, having progressed through the problems section, spotted an error in one of the slides. The student went on to hypothesise, correctly, about what the slide should say. This demonstrates that the problems section was readily absorbed and understood by the student.

Four out of five students were able to use the virtual laboratory through the use of only the worksheet and helper pages provided. This demonstrates that there is adequate information to enable students to perform tasks in the virtual laboratory. However, more information can be provided in the helper documentation especially with respect to more advanced settings and simulations available.

6.13.2 Structured Feedback

There were three forms of evaluation used, a System Usability Scale (appendix 1.5), Perceived Educational Value (appendix 1.6) and a more general feedback form with open ended questions (appendix 1.11). The SUS results are shown in full in Table 28 and summarised in Table 29.

Question / Student ID	1	2	3	4	5
I enjoyed using this system	4	4	4	4	4
I found the system unnecessarily complex	2	2	2	1	2
I thought the system was easy to use	4	4	4	4	5
I think that I would need the support of a technical person to be able to use this system	1	3	2	2	1
I found the various functions in this system were well integrated	4	4	5	3	4
I thought there was too much inconsistency in this system	1	2	1	1	2
I would imagine that most people would learn to use this system very quickly	4	4	5	4	5
I found the system very cumbersome to use	2	2	2	1	1
I felt very confident using the system	3	4	4	4	5
I needed to learn a lot of things before I could get going with this system	2	2	1	1	2

Table 28: Full System Usability Scale results showing the question and individual student response where 1 is strongly disagree and 5 is strongly agree

The results shown in Table 28 is summarised in Table 29.

Student ID	SUS Score as a Percentage
1	77.5
2	72.5
3	85
4	82.5
5	87.5

Table 29: Summary of the results from an evaluation session using the System Usability Scale, the average score was 81%

The usability feedback from users shows all of the students were very comfortable with using the system. The overall average results are the highest results received in a usability test in this dissertation.

Question / Student ID	1	2	3	4	5
I feel that I have learned something by using this system	4	5	4	4	4
The wireless simulation provides believable information	4	5	5	5	4
I found it easy to find out information about the MAC layer protocols	3	5	5	5	5
The quality of the material presented was consistent	4	5	5	5	4
Reviewing the hidden node terminal problem helped me to understand it	4	5	4	5	5
I feel that using this system helps develop my understanding of wireless protocols	4	5	5	5	4
I found the system educationally stimulating	4	5	5	5	4
I was able to easily create different scenarios	4	5	4	5	5
The tools provided by the system allowed me to practice the theory that I have learned relating to wireless networks	4	5	5	5	4
The animation was intuitive and easy to understand	4	4	4	4	4

Table 30: Full perceived educational value responses from individual students where 1 is strongly disagree and 5 is strongly agree

The results shown in Table 30 are summarised in Table 31.

Student ID	Average Perceived Educational Value as Percentage
1	78
2	98
3	92
4	96
5	86

Table 31: Summary of the feedback from users on the Perceived Educational Value, the average score was 90%

The user was provided with four questions which enabled an open anonymous response. These questions are summarised here and available as they were supplied to the students in appendix 1.11.

1. Is there any aspect of the system you particularly liked?
2. Was there anything you found difficult to do in the system?
3. Are there any features not already present you would like to see added?
4. Do you have any comments about this session in general?

The comments received were very positive, with students highlighting parts of the laboratory session they enjoyed and suggesting improvements in other areas. An example of the positive feedback is demonstrated in Table 32.

1. I liked the slides and simulations available, they were easy to use and understand
2. It was difficult to remember the colour coding during the simulation
3. Some sort of portable colour coded chart for the simulation (object you can drop on the ground in front of simulation)
   Prevent users from deleting the slides
4. It was fun and interesting.

Table 32: Example open feedback from students

Most students commented that they enjoyed using the system as shown in the example feedback provided in Table 33.

- Other than some underlying parts, bugs in the world simulator for example, the session was quite effective
- It was fun and interesting
- An enjoyable experience overall, OpenSim was a bit buggy

Table 33: Selection of user feedback to Question 4 on the open feedback form

6.13.3 Informal Unguided Interview

The users expressed the view that the session was very enjoyable and that they could see many possibilities with a WiFi island. All users did say that they felt they had learned something through using the software. An interesting discussion with a student was the number of students on an island. They expressed concern that having 30 students on an island would cause great difficulty with activities such as reviewing lecture slides. The student, when told that they could have an island of their own, responded that this was not ideal either as they found it valuable to learn from seeing and interacting with what others were currently doing on the island.

6.14 *WiFiOS Evaluation Session 2*

A second evaluation session was performed using the WiFi Open Simulator instance in April 2010. The session was carried out as part of the curriculum for an honours degree in Computer Science at the University of St Andrews. The class consisted of 16 honours students in their third year of study.

A single OAR file was created that represented every asset on the virtual island. This OAR file was deployed onto a total of four islands. Each island was identical in content but had a slightly different name. There were a total of 16 students registered for the course. Each student had an *avatar* created for them which consisted of their forename and surname. This made *in-world* identification easy for both the students and the laboratory assistants. Students were given no formal introduction to virtual worlds or the clients that were to be used . A worksheet was provided that detailed how to log in to the virtual world and the questions to be answered. The worksheet was kept the same as in the previous evaluation session and is available in appendix 1.9. After the session, evaluation sheets were provided and the results are detailed in the following sections.

6.14.1 Observer Notes

The cohorts of students were very interested in the technology and the material provided. Most of the students involved stayed well beyond the required timeframe assigned for the laboratory session. There were active discussions on the hidden node and exposed node scenarios between students and laboratory assistants. Some students were able to quickly make their own scenarios without any help from assistants or other students.

As in previous laboratory sessions vandalism of the island was quickly performed. For the session, all settings restricting ownership and the ability to move or edit objects had been removed. The environment was, should they choose, possible to destroy. While buildings and lecture slides were deleted, this was only prevalent at the start of the session. Due to the ease of redeploying the island using an OAR file any damaged islands were returned to their initial state in under a minute. The vandalism was not a recurrent theme of the session and the students' attention was quickly turned to the wireless material.

6.14.2 Structured Feedback

The results of the Perceived Educational Value questionnaire are shown in Table 34. These results show that students rated the use of the WiFiOS highly and felt they had learned something from the system. The feedback sheet for this section is shown in appendix 1.6.

Student ID	Average Perceived Educational Value as Percentage
1	88
2	60
3	80
4	82
5	96
6	80
7	71
8	70
9	86
10	92

Table 34: Results from the perceived educational value from 10 students. The average score was 81%

The students also evaluated the usability of the system using the System Usability Scale (appendix 1.5), the results of which are shown in Table 35. The average score for usability was 77%, which is encouragingly high for a novel interface.

Student ID	SUS Score as a Percentage
1	85
2	68
3	83
4	83
5	90
6	63
7	65
8	56
9	83
10	95

Table 35: Summary of the results from students on the usability of WiFi Open Simulator, the average score was 77.1%

An open-ended feedback form was also supplied to students. This allowed the students to express their views in a semi-structured way. The questions were:

1. Is there any aspect of the system you particularly liked?
2. Was there anything you found difficult to do in the system?
3. Are there any features not already present you would like to see added?
4. Do you have any comments about this session in general?

In the proceeding tables direct quotes from the students feedback forms are provided.

- Good, realistic simulation of wireless protocols
- The whole idea of it is cool
- Real-time visual representation of the wireless scenarios were particularly useful to help understand them
- Interaction

Table 36: User feedback to the open-ended question 1 on aspects of the system they particularly liked

- Sometimes animations kick off before the slide becomes readable, preventing one to read the slide at all, as the bubbles hinder vision
- Prevent people stealing items I was working with!
- Navigate around different sections and knowing where they were
- Figure out where to go
- Go up stairs

Table 37: User feedback to the open-ended question 2 on anything they found difficult to do in the system

The user feedback shown in Table 36 provides encouraging results and shows that the students were engaged and enjoyed using the system. There are important problems raised in Table 37 with a

prominent one being difficulty in navigation and vandalism from other students on a shared resource. The central portion of the island does provide a way to navigate around the island using a teleportation command. Navigation around the island may also be fixed by increasing the draw distance in the viewer settings. The default value is to show only 64 metres. When this is increased to show the full island, students found navigation easier. The results from the WiFiOS evaluation sessions are summarised in Table 38.

WiFiOS	
Number of Participants	15
Evaluation Methodology	System Usability Scale, Perceived Educational Value, Unstructured Interviews, Structured Interviews and Direct Observation
System Usability Scale Average	78%
Perceived Educational Value Average	84%

Table 38 Summary of the evaluation methodology and results obtained during the WiFiOS evaluation

6.15 *Summary*

This chapter has detailed the progression of a virtual laboratory into an immersive virtual world. What has been shown is the initial design decisions required in creating an environment in the commercially restrictive Second Life. Having evaluated its use in Second Life, we were able to see clear benefits of using an Open Simulator installation to provide students with supporting materials and their own workspaces. This work provides details on the process of creating and evaluating a learning resource in two separate and competing virtual worlds. Through the use of Open Simulator, students can freely destroy and modify the environment and because of the ability to load an image of the island (OAR), this encourages exploratory learning. In laboratory sessions with users, it is clear they are interested and engaged in the material presented and this is reflected in the perceived educational value and system usability scores as well as the comments provided in the open-ended feedback forms.

7 Conclusion

7.1 *Introduction*

The challenges of supporting an exploratory approach to computer networking education include barriers of cost (equipment) and time (the busy curriculum). A simulation-based framework which accommodates usability heuristics was identified as a means of addressing those challenges.

This dissertation has described a framework which includes a powerful simulator and provides appropriate access to it in a virtual laboratory within an educational setting. The different technologies and approaches used in the creation of the system have been detailed. A simple 2-dimensional user interface, a web form, was developed with a feature set that is appropriate for undergraduates and taught postgraduates. This has been deployed and evaluated. Evaluation has shown that it has helped students engage with and better understand the IEEE 802.11 wireless protocol. An alternative user interface was then developed to exploit the potential for GUI style usability in web pages afforded by RIAs.

The WiFiVL framework was further developed to support a virtual laboratory in a 3-dimensional, immersive, virtual world. There have been several approaches considered in creating a 3-dimensional environment for education. Online multi-player computer games were considered as well as 3-dimensional modelling and virtual worlds. Each of these approaches differ from the web as they provide a new degree of freedom. They also offer a new possibility in presenting the visualisation of a simulation. Virtual worlds allow different perspectives that are not possible in the 2-dimensional realm. Navigation in a 3-dimensional space is potentially intuitive as that is how we live our lives.

Virtual worlds enable enhanced presence as each user is represented by their avatar in the same, shared locale. This is very useful for group work and is accomplished in a unique and engaging way in 3-dimensional environments.

There were two main virtual worlds considered for use in this dissertation: Second Life and Open Simulator. Second Life is a mature commercial service with a very large global user base. Open Simulator is an open source project that is still in its early stages but provides full access to the software base needed for running an island in a self-hosted virtual world. Significantly Open Simulator enables the developer to manage a local installation of an island without any recurring cost. Programming in Second Life is designed to protect other users and scripts from your program, ensuring fairness and combating malicious use. However, it is also very restrictive with respect to communications with systems outside of Second Life. These issues are removed to a large extent when the programmer can manage their own private island in Open Simulator.

The virtual world implementation created a setting that could pull together many different types of learning interaction from traditional text, posters, streamed lectures and the novel 3-dimensional virtual laboratory where nodes can be manifested as 3-dimensional representations of laptops and base

stations. The students were provided with an engaging environment where they could interact with each other and the resources provided to them. The use of virtual worlds and the resources created were reviewed by the students and has shown to be engaging and have perceived educational value.

As both 2-dimensional and 3-dimensional versions of WiFiVL were created it is possible to provide a brief comparison of the different interfaces. This dissertation presents a strong case for the benefits of virtual world learning due to the level of engagement and the ability for collaborative work. The benefits of 2-dimensional interfaces are that they can be easier to develop and have a lower learning curve for users.

A summary of all the evaluation sessions is provided in Table 39. Table 39 shows that the Perceived Educational Value increases with the SUS. It also shows that the progression of interfaces has been valued by students.

	Number of Participants	Average System Usability Score	Average Perceived Educational Value score	Interviews
WiFiVL I	22	63	63	
WiFiVL II	16	76	76	
WiFiSL	3	78	78	✓
WiFiOS	15	78	84	✓

Table 39: Summary of the evaluation results across all versions of the WiFi Virtual Laboratory

There are aspects of this dissertation which could be further developed, given time, and these are highlighted in the future work section.

7.2 *Future Work*

The current educational approach to using a simulator is to construct a scenario that involves traffic for a finite time between two nodes and then simulate that result in a simulator, the output of which is animated as a series of events. An expansion of this would be to show an ongoing simulation where actions can be taken as the simulation is in progresses. For example, a two node scenario where a simple FTP communication is taking place, the student then moves a concrete wall into place between the nodes. The loss of communication should be immediately evident and were there to be more nodes placed around the area then you could observe the routing protocols renegotiating.

This project could be enhanced through incorporating ns-2, or an alternative simulator, as an actual service within the source code of the virtual world. This could allow individual students to script their own simulations through a set API or provide quicker and easier to maintain source code for WiFiSL.

Whilst the simulator used in this project is ns-2 and the protocol explored is the IEEE 802.11 the design is such as to allow easy adaptation to other simulators or emulated labs. The paper in [303] uses Field Programmable Gate Arrays (FPGA's) and wireless cards for six wireless stations. Using the FPGA's it is possible to specify different receiving strengths between two stations; this would allow the emulation

of a wireless scenario of six nodes distributed about an island with objects between certain nodes. Results from the emulation could be displayed back to the users in a virtual world using the same protocol developed for WiFiSL.

Other types of scenarios could also be explored and created. The virtual world provides easy manoeuvrability of the *avatar* and other objects in the world. Because of this manoeuvrability mobility of nodes is an interesting area for consideration. The mobility of nodes allows the exploration of the *disassociation* and *reassociation* of nodes with base stations.

The ability to explore the use and functionality of Access Points is very relevant given their prevalence in modern society. Access Points provide a central point for a group of nodes to coordinate. Access Points can be linked together to provide a more substantial infrastructure that link smaller ad hoc networks to larger or wired networks. Section 4.10 shows that Access Points provide a range of functionality and their inclusion in scenario construction in WiFiSL enables a widespread range of scenarios to be considered by students. This component is also represented as an asset in their *inventory*, which can be deployed inside the MUVE. Specific Access Point functionality such as *association* can be toggled on or off and the resulting messages between nodes and Access Points reviewed. The interactions between multiple Access Points are another scenario for exploration.

An example scenario utilising an Access Point is where multiple groups of nodes are deployed, each of which associates with a central Access Point. This structure can be repeated over an area incorporating different sizes of associated nodes. The instantiation of a traffic connection between two endpoints outside of each others cluster would allow the exploration of access point interaction. Whilst simulations can be constructed with Access Points this has not been focused on in worksheets or extensively tested in evaluation sessions with students.

An alternative approach to simulating in virtual worlds is to write the simulator in the native language of that world, in the case of Second Life that would be in LSL. The simulator could be of a similar design to Jasper as discussed in section 3.2 which is a finite state machine with clear transitions between states. Writing a Jasper-style simulator would be simpler to write in LSL than a port of the powerful but complex ns-2.

7.3 *Conclusion*

This dissertation has reported an encompassing study of creating virtual laboratories for exploratory learning about wireless networks in 2 and 3 dimensions. It has been demonstrated that there is a great interest in students to engage with and experiment in virtual laboratories for education. The dissertation provides a review of the transposition from traditional web-based interfaces to novel and experimental virtual worlds. The virtual worlds explored include both commercial (Second Life) and an open source alternative (Open Simulator). What has ultimately been shown is that students are keen to experiment with educational topics in modern and engaging formats and this has been supported by the WiFiVL framework.

1 Appendix

1.1 *Virtual World Glossary*

- **Asset** – Used to refer to any item in a virtual world, this includes primitives, scripts and note cards.
- **Avatar** - An *avatar* is a representation of the user in the virtual world.
- **Coordinate System** – An island is 256 (virtual) metres wide and long with essentially unlimited height. Each point on the island can be referenced using its x, y and z coordinate.
- **Draw Distance** – This is the maximum distance at which objects will be rendered. This does not include surrounding features such as the land, sea or air which are designed to look all-encompassing.
- **Inventory** – The *inventory* is a place for keeping virtual assets which can be deployed into the world through drag and drop. Both the user and primitives have an *inventory*, allowing prims to store other prims and scripts.
- **Linden Scripting Language (LSL)** – LSL is the scripting language used in Second Life which is compiled to Mono [304] (an open source implementation of .NET framework). The scripting language is well documented [223] and stable providing a wide range of capabilities. Scripts are associated with primitives and have many states that are triggered through user interactions or other scripts.
- **Notecard** – A text document that is stored in the inventory. Often used to describe a place or an object and can be given to other avatars.
- **Rezzing** – This is a process where new primitives can be added to the virtual world. New primitives can be added either by using the software viewers build interface or a script can trigger the addition of another new primitive. If a script is to add a primitive then it must already be a part of the *inventory* of the object hosting that script.
- **Primitives (Prims)** – Primitives are the building blocks of the virtual world and are available in a range of shapes e.g. a sphere, cube and torus. Each primitive has its own inventory where any asset can be stored.
- **Viewer** – "Second Life Viewer" is the software used to observe and interact with the virtual world [305].

1.2 *Laboratory Exercise sheet provided to Master students at the University of St Andrews*

CS5021 Laboratory Exercise Sheet #1 Understanding WiFi

Learning Outcomes:

- familiarity with the 802.11 (WiFi) protocol

- familiarity with the evolution of the ethernet group of 802.xx LANs

- understanding of the rationales behind the 802.11 protocol design

- experience of using the WiFi virtual laboratory

- experience of using a WiFi monitoring tool

- experience of setting up a WiFi router

1. Open a Word document (or similar) and be prepared to answer questions as they occur in the exercises.

2. Go to **http://wifi.cs.st-andrews.ac.uk**

a. Navigate to the 802.11 frame format diagram.

i. Why are there 4 address fields? Which other frame components are associated with the use of all four addresses?

ii. What types of frame are possible?

iii. Which frame components support fragmentation? What is the rationale behind fragmentation?

iv. What is CSMA/CA and why was it selected as a MAC policy for WiFi?

b. 802.11 can be seen as a recent stage in the evolution of LAN technologies and protocols. In what main ways does it differ from ALOHA and Ethernet (802.3) with respect to i) reliability and ii) security?

3. Go to **either http://wifi.cs.st-andrews.ac.uk and enter the virtual lab section or try the latest version at http://distsyst-dev.cs.st-andrews.ac.uk/WamNetMod/**

a) Try creating some scenarios (see the Primer); this can be saved by copying and pasting into a text document.

b) Two distinctive problems encountered in wireless communication are the hidden node and the exposed node scenarios. Study the information content on the WiFiVL site on these problems.

Use textbooks and discussions with your classmates and to develop further understanding.

Once you have understood these problems try creating instances of them using the WiFiVL. This will require using appropriate grid co-ordinates and sensing strength. Pay particular attention to the use of RTS/CTS exchanges when the simulations are running. Save the URLs for your scenarios in the lab session report. Explain in detail why you set the parameters to values you have chosen to simulate the scenarios.

4. Identify the MAC address of your MacBook. What is it? Download and install Kismac on the MacBook. Familiarise yourself with the application. Briefly, what information can it discover about wireless networks and traffic? Give a list of some of the base station Mac addresses associated with Portable.DCS, and which parts of the Computer Science buildings they serve.

5. Set up your MacBook as a router. This requires connecting to the network via the Ethernet interface cable then advertising a new wireless network. Work with someone else to discover which IP address ranges the temporary wireless network is allocating, and include these in your report.

Upload your report to MMS within 7 days of the supervised laboratory session.

1.3 *Laboratory Exercise sheet provided to CS3102 students at the University of St Andrews*

CS3102 WiFiVL Laboratory Exercise Sheet

Learning Outcomes:

familiarity with the 802.11 (WiFi) protocol

familiarity with the evolution of 802.xx LANs

understanding of the rationales behind the 802.11 protocol design

experience of using the WiFi virtual laboratory

Open a Word document (or similar) and be prepared to answer questions as they occur in the exercises. Index your answers using the numbering scheme e.g. 2.a.ii.

Go to **http://wifi.cs.st-andrews.ac.uk**

Navigate to the 802.11 frame format diagram.

Why are there 4 address fields? Which other frame components are associated with the use of all four addresses?

What types of frame are possible?

Which frame components support fragmentation? What is the rationale behind fragmentation?

What is CSMA/CA and why was it selected for WiFi?

b. 802.11 can be seen as a recent stage in the evolution of LAN technologies and protocols. In what main ways does it differ from i) ALOHA, ii) Ethernet (802.3)? e.g. How secure is 802.3? How secure is 802.11?

3. Go to **http://wifi.cs.st-andrews.ac.uk and enter the virtual lab section.**

a) Try creating some scenarios (see the Primer); this can be saved by copying and pasting into a text document.

b) Two distinctive problems encountered in wireless communication are the *hidden node* and the *exposed node* scenarios. Study the information content on the WiFIVL site on these problems. Use textbooks and discussions with your classmates and to develop further understanding.

Once you have understood these problems try creating instances of them using the WiFiVL. *This will require using appropriate grid co-ordinates and sensing strength.* Pay particular attention to the use of RTS/CTS exchanges when the simulations are running.

Upload your report to MMS

1.4 *Document provided to students to enable the use of the new interface*

Go to http://www.cs.st-and.ac.uk/~tommy/wifiui/WFEF.html

Create a scenario

To add a node click on the computer in the panel on the left hand side then click anywhere in the white area

To see if the nodes are in transmission distance of each other enable “show radius” toggle button

To add multiple nodes hold down ctrl while you click

To get your pointer back click on the black pointer

To create a connection between two nodes click on the source node, hold down shift and click on the destination node, a traffic connection dialogue will appear

The advanced options contains settings for MAC protocol, routing protocol and what packet size to use RTS/CTS for. Experiment with these values once you have a working simulation.

When you are finished setting up your scenario click “Launch” and wait for the XML file to download

Review the results of your scenarios simulation

The nodes will start communicating using flashing circles to indicate a wireless signal

The signals are colour coded with the legend at the top of the page

Text feed back is provided on the left hand side

Navigation through the scenario can be done using the event timebar

Analyse your simulation

Click on the “Stats for Sim” button to launch a page containing statistics for the scenario

Compare the impact of changing the MAC protocol, traffic density on the number of dropped packets and the distribution of packet types at the MAC level

Each WiFiVL scenario is described as a URL, by changing various parameters in the URL you will not have to reconfigure your scenario, here is a few hints:

&mp=11 This is the 802.11 protocol

&mp=ALOHA This is the Aloha protocol

&mp=tdma Time Division Multiple Access

To restart reload the page

1.5 *Questions from the System Usability Scale*

Question	Strongly disagree				Strongly agree
I think I would like to use this system frequently	1	2	3	4	5
I found the system unnecessarily complex	1	2	3	4	5
I thought the system was easy to use	1	2	3	4	5
I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
I found the various functions in this system were well integrated	1	2	3	4	5
I thought there was too much inconsistency in this system	1	2	3	4	5
I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
I found the system very cumbersome to use	1	2	3	4	5
I felt very confident using the system	1	2	3	4	5
I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

1.6 *Questions from the Educational Questionnaire*

Question	Strongly disagree				Strongly agree
I feel that I have learned something by using this system	1	2	3	4	5
The wireless simulation provides believable information	1	2	3	4	5
I found it easy to find out information about the MAC layer protocols	1	2	3	4	5
The quality of the material presented was consistent	1	2	3	4	5
Constructing and playing a simulation of the hidden terminal problem helped me understand it	1	2	3	4	5
I feel that using this system helps develop my understanding of wireless protocols	1	2	3	4	5
I found the system educationally stimulating	1	2	3	4	5
I was able to easily create different scenarios	1	2	3	4	5
The tools provided by the system allowed me to practice the theory that I have learned relating to wireless sensor networks	1	2	3	4	5
The animation was intuitive and easy to understand	1	2	3	4	5

1.7 *Task List*

Task Name	Status at end of session
Deploy networking components into the environment	
Construction of a scenario involving two nodes transferring a file	
Alter the routing protocol	
Alter the MAC protocol and describe the changes observed	
Modify the sensing range and observe the difference in	
Modify the RTS/CTS threshold and describe, with usage of supplied statistics, any changes observed	

1.8 *Laboratory Exercise sheet provided to Master students at the University of St Andrews*

CS5021 Laboratory Exercise Sheet #1

Understanding WiFi (IEEE 802.11)

Learning Outcomes:

- familiarity with the 802.11 (WiFi) protocol

- familiarity with the evolution of the ethernet group of 802.xx LANs

- understanding of the rationales behind the 802.11 protocol design

- experience of using the WiFi virtual laboratory

- experience of using a WiFi monitoring tool

1. Open a Word document (or similar) and be prepared to answer questions as they occur in the exercises. Index your answers using the numbering scheme e.g. 2.a.ii.

2. Go to http://wifi.cs.st-andrews.ac.uk

a. Navigate to the 802.11 frame format diagram.

i. Why are there 4 address fields? Which other frame components are associated with the use of all four addresses?

ii. What types of frame are possible?

iii. Which frame components support fragmentation? What is the rationale behind fragmentation?

iv. What is CSMA/CA, how does it contrast with CSMA/CD, and why was it selected as a MAC policy for WiFi?

v. What is the RTS/CTS threshold and how might it impact on CSMA/CA?

b. 802.11 can be seen as a recent stage in the evolution of LAN technologies and protocols. In what main ways does it differ from ALOHA and Ethernet (802.3) with respect to i) reliability and ii) security?

3. The site at http://wifi.cs.st-andrews.ac.uk has two types of components: informational hypermedia and a virtual laboratory. There are two interfaces to the virtual laboratory: forms-based and "point and click" style. The respective URLS are:

http://wifi.cs.st-andrews.ac.uk:8080/WamNetMod/

http://wifi.cs.st-andrews.ac.uk:8080/WamNetMod/WFEF.html

a) Try creating some scenarios; the simulations generated by the scenarios have URLs pointing to their XML files that can be copied for future reference, and entering into your Lab Report. Note that the "Stats for Sim" button launchs a page containing statistics for the scenario. These must ne intertepresetd carefully. They will include information on dropped frames.

b) Two distinctive problems encountered in wireless communication are the hidden node and the exposed node scenarios. Study the information content on the WiFiVL site on these problems.

Use textbooks and discussions with your classmates and to develop further understanding. Once you have understood these problems try creating instances of them using the WiFiVL. This will require using appropriate grid co-ordinates (forms-based interface only) and sensing strength. Pay particular attention to the use of RTS/CTS exchanges when the simulations are running. Note that the advanced options allow for setting a RTS/CTS threshold. This is the value of frame size above which CTS/RTS is used. What would you expect to happen to packet transmissions as the threshold is increased?

Save the URLs for your scenarios in the lab session report. Explain in detail why you set the parameters to values you have chosen to simulate the scenarios.

4. Identify the MAC address of your MacBook. What is it? Download and install Kismac and Wireshark on the MacBook. Help will be given on how to do this. Note that you may have to use "sudo" to enable Wireshark's traffic capturing capabilities). Familiarise yourself with these applications. What information can they discover about wireless networks and traffic? Give a list of some of the base station Mac addresses associated with Portable.DCS, and which parts of the Computer Science buildings they serve.

5. Read the draft paper (on studres) "Link Assessment in an Indoor 802.11 Network" by Michael R. Souryal, Luke Klein-Berndt, Leonard E. Miller, and Nader Moayeri. In not more than two sides of A4 explain which features of 802.11 the paper sets out to test, and summarise the results.

1.9 *Worksheet for Wifi OpenSim*

WiFi Virtual Laboratory Island Worksheet

Navigating to the OpenSim server Mimuve and the Island Aeolus

Start SecondLife client with the following command added:

--loginuri http://minerva.cs.st-andrews.ac.uk:8002

Use the supplied username and password

Once on the island try flying around and ensure you are capable in operating your avatar in a 3-dimensional environment

Start off in the Lecture Theatre and review some of the lecture material

What is CSMA and how does it function?

What is Aloha? Discuss the scalability of such a protocol.

Review the “Problems and Solutions in 802.11” section of the Island

What is the hidden node problem?

What is the exposed node problem?

How can these problems be solved?

What is the mechanism for data transfer in 802.11?

Review the “Frame Structure” section of the island and the 802.11 official protocol.

What types of frames are possible?

Why are there four address fields?

How many bits are there in the Frame Control field and what are some of its functions?

Which frame components support fragmentation? What is the rationale behind fragmentation?

What is the use of Seq-ctl?

Review the “WiFi Virtual Laboratory” section of the island

Construct and review the mechanism for transferring data

This will involve acquiring the appropriate equipment

Signalling a traffic connection between two nodes

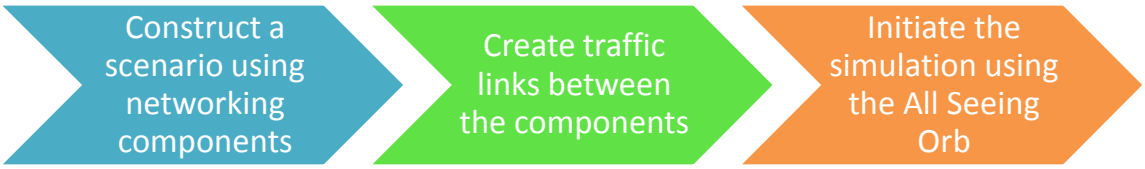
Sending the scenario off for simulation and the results will then be animated back

Advanced settings can be modified through the Orbs Notecard, instructions are included. Construct some scenarios and review the statistics generated. What factors can be modified to alter the number of dropped packets? How does this relate to section 2 and 3?

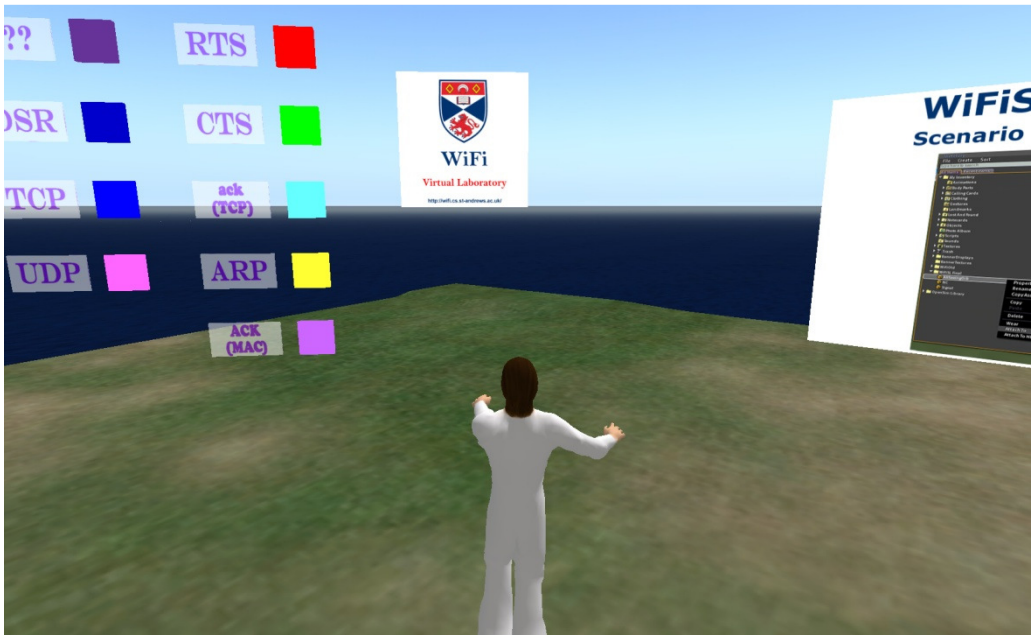
1.10 *WiFiVL User Guide for use in Open Simulator*

Guide to the WiFi Virtual Laboratory

The WiFiVL consists of three actions – constructing a scenario, simulating the scenario and reviewing the animation provided. These steps are summarised below and full details on how to use the WiFi Virtual Laboratory (WiFiVL) tool are contained in this document.



Click on the WiFiVL box, a L$ sign should appear and you can now purchase all the components required for no cost.



Three items should now be in your inventory:

Networking Component (NC) – This is a networking component and is used to represent a node in a network. This can be deployed anywhere throughout an island. Use the NC to construct traffic links which will be simulated.

All Seeing Orb (ASO)– This item gathers all the information from the NC's you have deployed throughout the island. This item should be attached to your body by right clicking on the item in the inventory and choosing attach to. You can initiate the simulation by touching this orb or by saying *wifirun* in the chat console.

SpeedControlBoard – Pull this board out if you want to change the speed of the simulation. As the TCP increases the frame size there may be periods of inactivity, to jump to the next series of interactions press the skip button.

How to Construct a Scenario

Attach the ASO to your avatar

Deploy the NC into the environment

Touch one node to indicate sender and a second node, within 2 seconds, to indicate receiver

Touch the ASO to trigger the simulation

Advanced Options

Use the WiFi command line to link together nodes, type help to see the options

Edit the Notecard in the ASO to change advanced options

1.11 *Open Feedback Form*

Question	Response
Is there any aspect of the system you particularly liked?	
Was there anything you found difficult to do in the system?	
Are there any features not already present you would like to see added?	
Do you have any comments about this session in general?	

References

[1] J. Kurose, et. al., *Computer Networking a Top-Down Approach*, 4th ed.: Addison-Wesley, 2007.

[2] D. E. Comer, *Computer Networks and Internets*: Prentice Hall, 2008.

[3] L. Peterson, *Computer Networks: A systems Approach*: Morgan Kaufmann, 2003.

[4] A. Tanenbaum, *Computer Networks*, 4 ed.: Prentice Hall, 2002.

[5] C. t. t. N. SAMAN (supported by DARPA), ICIR, Sun Microsystems, UCB Daedelus and Carnegie Mellon Monarch projects, "Ns-2. The Network Simulator," ed, 1988.

[6] JISC. (2007, 30/09/09). *e-Learning Focus - Technology Enhanced Learning Environments.* Available: http://www.elearning.ac.uk/subjects/PLE

[7] TELS. (2008, 29/09/09). *Technology Enhanced Learning in Science*. Available: http://telscenter.org/

[8] J. T. Robert McCartney. (2010, 25/05/10). *The ACM Transactions on Computing Education*. Available: http://toce.acm.org/

[9] Joint Information Systems Committee. (2010, 25/05/10). *JISC: Supporting education and research*. Available: http://www.jisc.ac.uk/

[10] Intel. (2005, *Excerpts from A Conversation with Gordon Moore's Law*. Available: ftp://download.intel.com/museum/Moores_Law/Video-Transcripts/Excepts_A_Conversation_with_Gordon_Moore.pdf

[11] R. Weitbrecht, "WEITBRECHT FREQUENCY-SHIFT TELETYPEWRITER," U.S.A Patent 3507997, 1970, 1966.

[12] D. R. Ken Thompson, Brian Kernighan, Douglas McIlroy, Joe Ossanna, "Unix," ed: AT&T, 1969.

[13] R. Molich, Nielsen, J., "Improving a human-computer dialogue," *Communications of the ACM,* vol. 33, pp. 338-348, 1990.

[14] Microsoft. (2007, 01/08/10). *Windows Vista (6.0 Service Pack 2 (SP2) ed.)*. Available: http://windows.microsoft.com/en-US/windows-vista/products/home

[15] Ubuntu Foundation. (2004, 01/08/10). *Ubuntu (9.04 ed.)*. Available: http://www.ubuntu.com/

[16] Apple Inc. (2009, 01/08/10). *Mac OS X (Snow Leopard 10.6 ed.)*. Available: http://www.apple.com/macosx/

[17] R. B. Roy Trubshaw, "MUD (Multi-User Dungeon)," ed, 1978.

[18] P. G. David Barham, John Medhurst, Dave Morris, Shaun Plumb, Paul Friday, Michael Lawrie, Bret Giddings, Richard Thombs, Adam Bird, Simon Smith, "MIST," ed: Essex University, 1987.

[19] M. S. Sebastian Hammer, Hans Henrik Staerfeldt, Tom Madsen, Katja Nyboe, "DikuMUD," Alpha ed, 1991.

[20] N. Newell, "SHADES ", ed, 1985.

[21] Xerox. (2009, 03/10/09). *Milestones - PARC (Palo Alto Research Center)*. Available: http://www.parc.com/about/milestones.html

[22] D. Weatherall, "OTcl (MIT Object Tcl)," ed, 1995.

[23] K. C. JB Biggs, *Evaluating the Quality of Learning: The SOLO Taxonomy (Structure of the Observed Learning Outcome)*: Academic Press, 1982.

[24] C. Linnaeus, *Systema Naturae*, 10 ed., 1735.

[25] B. Bloom, "Taxonomy of educational objectives: The classification of educational goals," *New York: David McKay,* 1956.

[26] E. Simpson, "THE CLASSIFICATION OF EDUCATIONAL OBJECTIVES, PSYCHOMOTOR DOMAIN," 1966.

[27] A. Harrow, *A taxonomy of the psychomotor domain: A guide for developing behavioral objectives*: Addison-Wesley Longman Ltd, 1972.

[28] R. Dave, "Developing and writing behavioural objectives," *Educational Innovators, Tuczon, Ed. Anderson,* 1975.

[29] L. Anderson*, et al.*, *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*: Longman, New York, 2001.

[30] D Fisher. (2007, 25/08/09). *Models -- Instructional Design The Taxonomy Table -- Faculty Resources -- OSU Extended Campus -- Oregon State University*. Available: http://oregonstate.edu/instruct/coursedev/models/id/taxonomy/
[31] *Theory into Practice* vol. 41: Lawrence Erlbaum Associates, 2002.
[32] C. Ferguson, "Using the revised taxonomy to plan and deliver team-taught, integrated, thematic units," *Theory into Practice,* pp. 238-243, 2002.
[33] F. Ursula*, et al.*, "Developing a computer science-specific learning taxonomy," presented at the Working group reports on ITiCSE on Innovation and technology in computer science education, Dundee, Scotland, 2007.
[34] L. S. Jimmy Wales. (2001, 11/06/07). *Wikipedia, the free encyclopedia*. Available: http://en.wikipedia.org/
[35] Wikimedia. (2009, 03/12/09). *Wikimedia Foundation*. Available: http://wikimediafoundation.org
[36] J. Schachter. (2003, 22/04/09). *delicious*. Available: http://delicious.com
[37] A. O. Steve Huffman. (2005, 24/08/09). *reddit.com: what's new online!* Available: http://www.reddit.com/
[38] G. E. Company. (2009, 24/08/09). *Imagination Cubed*. Available: http://www.imaginationcubed.com/index.php
[39] Microsoft, "MSN Messenger," Windows Live Messenger Build 8.1.0178.00 ed, 1999.
[40] Skype Limited, "Skype," 4.0.0.227 ed, 2003.
[41] P. Dillenbourg, *Collaborative Learning: Cognitive and Computational Approaches. Advances in Learning and Instruction Series*, 2 ed.: Elsevier Science, Inc., PO Box 945, Madison Square Station, New York, NY 10160-0757 ($72). Web site: http://www.elsevier.com., 1999.
[42] H. Srinivas. (2008, 24/08/09). *Collaborative Learning*. Available: http://www.gdrc.org/kmgmt/c-learn/
[43] P. Kirschner*, et al.*, "Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching," *Educational Psychologist,* vol. 41, pp. 75-86, 2006.
[44] M. Merrill, "A task-centered instructional strategy," *Journal of Research on Technology in Education,* vol. 40, p. 5, 2007.
[45] A. R. Fred Paas, John Sweller, "Cognitive Load Theory and Instructional Design: Recent Developments," *EDUCATIONAL PSYCHOLOGIST,,* vol. 38, pp. 1-4, 2003.
[46] B. Wadsworth, *Piaget's theory of cognitive and affective development: Foundations of constructivism*: Allyn & Bacon, 1995.
[47] E. von Glasersfeld, "A constructivist approach to teaching," *Constructivism in education,* vol. 3, p. 15, 1995.
[48] J. Rawls, "Kantian constructivism in moral theory: The Dewey lectures 1980," *Journal of Philosophy,* vol. 77, p. 72, 1980.
[49] S. Carey and R. Gelman, *The epigenesis of mind: Essays on biology and cognition*: Lawrence Erlbaum, 1991.
[50] L. Vygotsky, "Mind and society: The development of higher mental processes," ed: Cambridge, MA: Harvard University Press, 1978.
[51] S. Papert, *Mindstorms: Children, computers, and powerful ideas*, 1980.
[52] LEGO. (2009, 25/08/09). *LEGO.com MINDSTORMS NXT Home*. Available: http://mindstorms.lego.com
[53] E. Ackermann, "Piaget's constructivism, Papert's constructionism: What's the difference," *Future of learning group publication,* 2001.
[54] T. D. J. Melanie Njoo, "Exploratory learning with a computer simulation for control theory: Learning processes and instructional support," *Journal of Research in Science Teaching,* vol. 30, pp. 821-844, 1993.
[55] J. Rieman, "A field study of exploratory learning strategies," *ACM Transactions on Computer-Human Interaction (TOCHI),* vol. 3, pp. 189-218, 09/96 1996.
[56] J. Shrager, "Title," unpublished.
[57] M. Wertheimer, *Productive thinking*: Harper New York, 1959.
[58] C. Lewis, "Why and how to learn why: Analysis-based generalizations of procedures.," *Cog. Sci.,* vol. 12, pp. 211-256, April-June 1988 1988.

[59] M. Franzke, "Turning research into practice: characteristics of display-based interaction," in *Human Factors in Computing Systems*, Denver, Colorado, United States, 1995, pp. 421-428.
[60] *GNU GENERAL PUBLIC LICENSE,* GNU License, 2007.
[61] K. Turner. (2006, 16/02/07). *JASPER (Java Simulation of Protocols for Education and Research).* Available: http://www.cs.stir.ac.uk/~kjt/software/comms/jasper.html
[62] J. Lepreau. (2007, 16/08/08). *Emulab.Net - Emulab - Network Emulation Testbed Home*. Available: http://www.emulab.net/
[63] M. 3, "OPNET," ed.
[64] C. McDonald. (1996, 11/01/08). *The cnet network simulator (v2.0.10).* Available: http://www.csse.uwa.edu.au/cnet/index.html
[65] G. L. a. T. L. Jeff Chiang. (2005, 10/12/08). *WebLan-Designer Home*. Available: http://elena.aut.ac.nz/homepages/weblandesigner/
[66] C. B. A Ferrero, M Parmigiani. (2003, 16/11/09). *ReMLab*. Available: http://sermis.etec.polimi.it
[67] (2009, 15/09/09). *Sensasim Login Page*. Available: http://www.ait.edu.gr/ait_web_site/research_MKWT_projects.jsp
[68] L. DARPA, Xerox PARC, UCB, and USC/ISI. (2007, 12/06/08). *User Information - Nsnam.* Available: http://nsnam.isi.edu/nsnam/index.php/User_Information
[69] C. A. Tommy Sturgeon, Alan Miller, Clare Kerbey, Suzanne Hague (2007, 05/06/06). *WiFi Virtual Laboratory Primer.* Available: http://wifi.cs.st-andrews.ac.uk/
[70] D. G. Laurent, Jourjon Emmanuel, Lochin Sebastien, Ardon. (2008, 12/12/08). *IREEL.* Available: https://ireel.npc.nicta.com.au/experiments/auth/
[71] D. R. Jeremy Grossmann, Romain Lamaison, Aurélien Levesque, Xavier Alt (2008, 12/03/09). *News | GNS3*. Available: http://www.gns3.net/news/gns3-0.5-released.html
[72] S. Karlin. (2007, 01/12/07). *PlanetLab Design Notes*. Available: http://www.planet-lab.org/doc/pdn
[73] T. White. (2006, 28/07/10). *jFAST a Java Finite Automata Simulator* Available: http://www46.homepage.villanova.edu/timothy.m.white/
[74] Virtualsquare Team. (2007, 28/07/10). *VDE - Virtual Distributed Ethernet*. Available: http://vde.sourceforge.net/
[75] N. M. Martin Casado. (2010, *VNS*. Available: http://yuba.stanford.edu/vns/
[76] K. T. Iain Robin, "JASPER," 1.3 ed, 2001.
[77] K. T. Iain Robin, Paul Johnson, Kenneth Whyte. (2006, 20/05/10). *Protocol Simulators*. Available: http://www.cs.stir.ac.uk/~kjt/software/comms/jasper/
[78] I. A. R. J. Turner, "An interactive visual protocol simulator," *Computer Standards & Interfaces* vol. 23, pp. 279-310, 2001.
[79] J. D. Robert Ricci, Pramod Sanaga, Daniel Gebhardt, Mike Hibler, Kevin Atkinson, Junxing Zhang, Sneha Kasera, Jay Lepreau, "The Flexlab Approach to Realistic Evaluation of Networked Systems," in *Fourth USENIX Symposium on Networked Systems Design and Implementation (NSDI 2007)*, Cambridge, MA, 2007.
[80] T. S. David Johnson, Russ Fish, Daniel Montrallo Flickinger, Leigh Stoller, Robert Ricci, Jay Lepreau "Mobile Emulab: A Robotic Wireless and Sensor Network Testbed," presented at the Computer Communications (IEEE INFOCOM 2006), 2006.
[81] B. V. d. B. J. Theunis, P. Leys, J. Potemans1, E. Van Lil, A. Van de Capelle "Title," unpublished|.
[82] P. L. J. Theunis, J. Potemans1, B. Van den Broeck, E. Van Lil and A. Van de Capelle, "Title," unpublished|.
[83] C. M. Donald, "CNET - The cnet network simulator (v2.0.10)," ed, 1991.
[84] N. I. Sarkar, & Petrova, K., "WebLan-Designer: A Webbased system for interactive teaching and learning LAN design.," in *3rd IEEE International Conference on Information Technology Research and Education*, Hsinchu, Taiwan, 2005, pp. 328-332.
[85] C. B. A Ferrero, M Parmigiani, "ReMLab: A Java-based remote, didactic measurement laboratory," *IEEE Transactions on Instrumentation and Measurement,* vol. 52, pp. 710-715, 2003.
[86] A. Pierri*, et al.*, "Application of a virtual scientific experiment model in different educational contexts," *Exploiting the Knowledge Economy: Issues, Applications and Case Studies,* vol. 3, 2006.

[87] *GNU LESSER GENERAL PUBLIC LICENSE,* I. Free Software Foundation 3, 2007.
[88] D. G. Laurent, Jourjon Emmanuel, Lochin Sebastien, Ardon, "IREEL: remote experimentation with real protocols and applications over an emulated network," *SIGCSE Bull.,* vol. 39, pp. 92-96, 2007.
[89] D. G. Laurent, Jourjon Emmanuel, Lochin Sebastien, Ardon. (2008, *IREEL.* Available: https://ireel.npc.nicta.com.au/experiments/
[90] J. Dugan, "Iperf," 2.0.4 ed, 2008.
[91] Internet2, "Shibboleth - Web Single SignOn (SSO)," ed, 2007.
[92] R. Luigi, "Dummynet: a simple approach to the evaluation of network protocols," *SIGCOMM Comput. Commun. Rev.,* vol. 27, pp. 31-41, 1997.
[93] H. B. Michael Matthes, Michael Lauer, Oswald Drobnik, "MASSIVE: An Emulation Environment for Mobile Ad-Hoc Networks," presented at the Second Annual Conference on Wireless On-demand Network Systems and Services (WONS'05), 2005.
[94] Y. Z. a. W. Li, "An Integrated Environment for TestingMobile Ad-Hoc Networks.," in *In Proc. of the third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02)*, Lausanne, Switzerland, 2002.
[95] R. Russell, "iptables," 1.4.0 ed, 1998, p. Packet filtering.
[96] D. R. Jeremy Grossmann, Romain Lamaison, Aurélien Levesque, Xavier Alt "Graphical Network Simulator 3 (GNS3)," 0.5 ed, 2008, p. GNS3 is a graphical network simulator that allows you to design complex network topologies and to launch simulations on them.
[97] Cisco, "Title," unpublished|.
[98] "Dynamips - Cisco 7200 Simulator," 0.2.8-RC2 ed, 2007.
[99] G. Anuzelli, "Dynagen - The Network Configuration Generator for Dynamips," 4.0 ed, 2006, pp. Dynagen is a front-end for use with the Dynamips Cisco router emulator. .
[100] D. Bickson and D. Malkhi, "The julia content distribution network," in *WORLDS '05*, San Francisco, California, USA, 2005.
[101] L. Wang*, et al.*, "Reliability and security in the CoDeeN content distribution network," in *USENIX Annual Technical Conference*, 2004, p. 14.
[102] M. Castro*, et al.*, "Splitstream: High-bandwidth content distribution in cooperative environments," *Peer-to-Peer Systems II,* pp. 292-303, 2003.
[103] A. Sentinelli*, et al.*, "IPTV-P2P clients at home," in *16th International Conference on Systems, Signals and Image Processing*, 2009, pp. 1-4.
[104] B. G. Sean Rhea, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, Harlan Yu, "OpenDHT: A Public DHT Service and Its Uses," in *Applications, technologies, architectures, and protocols for computer communications*, 2005, p. 84.
[105] P. Wang*, et al.*, "A k-anonymous communication protocol for overlay networks," in *ASIAN ACM Symposium on Information, Computer and Communications Security*, 2007, pp. 45 - 56.
[106] PlanetLab Architecture Team, "Dynamic Slice Creation," 01-10-02 2002.
[107] A. M. C Allison, I Oliver, T Sturgeon, "Global Connections for Lasting Impressions: Experiential Learning about TCP," *Advances in Web Based Learning–ICWL,* pp. 48-57, 2009.
[108] T. W. Timothy White, "jFAST: a java finite automata simulator," presented at the Proceedings of the 37th SIGCSE technical symposium on Computer science education, Houston, Texas, USA, 2006.
[109] G. Michael and D. Renzo, *VDE: an emulation environment for supporting computer networking courses*. Madrid, Spain: ACM, 2008.
[110] C. Martin and M. Nick, *The virtual network system*. St. Louis, Missouri, USA: ACM, 2005.
[111] M. S. Krishna and V. Rajaravivarma, "Education of wireless and ATM networking concepts using hands-on laboratory experience," vol. 31, pp. 114-118, 1999.
[112] Y. H. Vasil and F. L. Andrea, "Undergraduate data communications and networking projects using opnet and wireshark software," presented at the Proceedings of the 39th SIGCSE technical symposium on Computer science education, Portland, OR, USA, 2008.
[113] The Wireshark Team, "Wireshark," 1.0.2 ed, 2008, pp. Cross-platform packet sniffing.
[114] J. Banks and J. S. Carson, *Discrete event system simulation*. United States: Prentice Hall,Englewood Cliffs, NJ, 1984.
[115] ETS. (2008, 27/08/08). *ETS: Educational Testing Service*. Available: www.ets.org

[116] E. Tamer and A. U. Shankar, "Using SeSFJava in teaching introductory network courses," presented at the Proceedings of the 36th SIGCSE technical symposium on Computer science education, St. Louis, Missouri, USA, 2005.
[117] F. D. M. Crescenzio, Giovanni Persiani, Franco Bombardi, Tiziano, "A First Implementation of an Advanced 3D Interface to Control and Supervise UAV (Uninhabited Aerial Vehicles) Missions," *Presence: Teleoperators & Virtual Environments,* vol. 18, pp. 171-184, 2009.
[118] T. M. M. Michael W. Wynne, William R. Looney III "AETC White Paper. On Learning: The Future of Air Force Education and Training," ed. United States Air Force - Air Education and Training Command, 2008.
[119] V. Kaptelinin and M. Cole, "Individual and collective activities in educational computer game playing," *CSCL 2, carrying forward the conversation,* p. 303, 2002.
[120] A. LEONTIEV, "Activity, consciousness, personality. Englewood Cliffs, N3: Printice Hall," 1978.
[121] N. Maxis, "Sim City," 1 ed: Electronic Arts, 1989.
[122] D. Gorlin, "Choplifter," ed: Ariolasoft, 1982.
[123] J. Clarke and C. Dede, "Making learning meaningful: An exploratory study of using multi-user environments (MUVEs) in middle school science," *AERA, Montreal,* 2005.
[124] C. Dede*, et al.*, "Designing for motivation and usability in a museum-based multi-user virtual environment," *Proceedings of AERA 2003,* 2003.
[125] C. D. DJ Ketelhut, J Clarke, B Nelson, C Bowman, "Studying situated learning in a multi-user virtual environment," *Assessment of problem solving using simulations. Mahwah, NJ: Lawrence Erlbaum Associates,* 2008.
[126] ISTE. (2009, 25/11/09). *ISTE Second Life*. Available: http://www.iste.org/Content/NavigationMenu/Membership/Member_Networking/ISTE_Second_Life.htm
[127] SimTeach. (2008, *Institutions and Organizations in SL*. Available: http://www.simteach.com/wiki/index.php?title=Institutions_and_Organizations_in_SL
[128] NOAA. (2008, 21/02/08). *NOAA (National Oceanic & Atmospheric Administration) Vritual World - Meteora*. Available: http://slurl.com/secondlife/Meteora/177/161/27/
[129] R. Schingler. (2008, 21/02/08). *Welcome to CoLab | NASA CoLab* [HTTP]. Available: http://colab.arc.nasa.gov/
[130] J. X. Y. Y. L. Chen, B. , "MUVEES: a PC-based multi-user virtual environment for learning," *Virtual Reality, 2003. Proceedings. IEEE,* pp. 163-170, 2003.
[131] M. Lafsky. (2009, Can Training in Second Life Teach Doctors to Save Real Lives? *Discover*. Available: http://discovermagazine.com/2009/jul-aug/15-can-medical-students-learn-to-save-real-lives-in-second-life
[132] B. N. Chris Dede, Diane Jass Ketelhut, Jody Clarke, Cassie Bowman, "Design-based research strategies for studying situated learning in a multi-user virtual environment," in *International Conference on Learning Sciences. Proceedings of the 6th international conference on Learning sciences*, Santa Monica, California, 2004, pp. 158-165.
[133] M. L. Jugel, "Enhancing MUVEs: Connecting virtual objects with environmental simulation," in *Proceedings of the International Conference on Virtual Worlds and Simulation*, Phoenix, Arizona, USA, 2000.
[134] A. L. JC Bradley. (2008, 12/08/09). *Drexel CoAS talks mp3 podcast: Chemistry Concepts in Second Life - Bradley/Lang*. Available: http://drexel-coas-talks-mp3-podcast.blogspot.com/2008/08/chemistry-concept-in-second-life.html
[135] Drexel University. (2007, 12/08/09). *Drexel Island*. Available: http://slurl.com/secondlife/Drexel/125/175/25
[136] Royal Society of Chemistry. (2007, 20/08/09). *ChemSpider*. Available: http://www.chemspider.com
[137] A. M. K Getchell, C Allison, C Kerbey, R Hardy, R Sweetman, V Crook, J Complin, "THE LAVA PROJECT," 2008.
[138] K. Aitchison, "Supply, demand and a failure of understanding: addressing the culture clash between archaeologists expectations for training and employment in 'academia' versus 'practice'," *World Archaeology,* vol. 36, pp. 203-219, 2004.
[139] Bytonic Software, "Jake2," 0.9.5 ed, 2006, p. Java port of the GPL release of Quake II.

[140] Sun. (2010, *Java SE Desktop Technologies - Java Web Start Technology*. Available: http://java.sun.com/javase/technologies/desktop/javawebstart/index.jsp
[141] C. Martin*, et al.*, "An approach to virtual-lab implementation using Modelica," 2006, pp. 137–141.
[142] Modelica Association. (2008, 01/05/09). *Modelica - Modeling of Complex Physical Systems*. Available: http://www.modelica.org/
[143] M. S. Richard and L. D. John, "Using DLSim 3: a scalable, extensible, multi-level logic simulator," presented at the Proceedings of the 13th annual conference on Innovation and technology in computer science education, Madrid, Spain, 2008.
[144] C. Burch, "A picocontroller training simulator in a Web page.," *International Journal of Electrical Engineering Education,* vol. 40, pp. 158-168, 2003.
[145] J. G. Jesús Adolfo García-Pasquel, "Ganzúa: A cryptanalysis tool for monoalphabetic and polyalphabetic ciphers," *Journal on Educational Resources in Computing (JERIC),* vol. 6, 2007.
[146] Y. Xiaohong*, et al.*, "An animated learning tool for Kerberos authentication architecture," *J. Comput. Small Coll.,* vol. 22, pp. 147-155, 2007.
[147] R. N. L. Carney, J.R., "Pictorial illustrations still improve students' learning from text," *Educational Psychology Review,* vol. 14, pp. 5-26, 2002.
[148] B. Weidenmann, *When good pictures fail: An information processing approach to the effect of illustrations*. New York, 1989.
[149] J. H. a. S. Larkin, H.A., "Why a diagram is (sometimes) worth ten thousand words," *Cognitive Science,* vol. 11, pp. 65-99, 1987.
[150] D. a. J. Bitzer, R., "PLATO: A Computer-based System Used in the Engineering of Education," *Proceeding of the IEEE,* vol. 6, pp. 960-968, 1971.
[151] L. R. A. Richard T. Murphy, "Evaluation of the PLATO IV computer-based education system in the community college," *ACM SIGCUE Outlook,* vol. 12, pp. 12-28, 01/01/78 1978.
[152] B. A. Sherwood, "The TUTOR language, Computer-Based Ed. Res. Lab," ed: U. of Illinois, Urbana-Champaign, IU., 1974.
[153] Z. U. Abbas, M. Odeh, M. McClatchey, R. Ali, A. Farooq, A. , "A semantic grid-based e-learning framework (SELF)," 2005, pp. 11-18.
[154] Open Source Applications Foundation, "Chandler," 0.7.3 ed, 2007.
[155] S. B. Shum, "CoAKTinG: Collaborative Advanced Knowledge Technologies in the Grid," in *Proc. Second Workshop on Advanced Collaborative Environments, Eleventh IEEE Int. Symposium on High Performance Distributed Computing (HPDC-11)*, Edinburgh, Scotland, 2002.
[156] S. B. S. A Slavin, M Sierhuis, J Conklin, "Compendium: Making meetings into konwledge events," in *Knowledge Technologies 2001*, Austin, Texas, USA, 2001.
[157] S. Tsekeridou*, et al.*, "Toward Virtual Campuses: Collaborative Virtual Labs & Personalized Learning Services in a Real-Life Context," in *Seventh IASTED International Conference*, 2008.
[158] IEEE, "IEEE 802.11 LAN/MAN Wireless LANS," ed, 1999.
[159] BT. (2009, 25/11/09). *BT Openzone*. Available: http://btopenzone.hotspot-directory.com/
[160] R. B. George Almási, "An Overview of the Blue Gene/L System Software Organization " in *Lecture Notes in Computer Science*. vol. 2790/2004, ed: Springer Berlin / Heidelberg, 2004, pp. 543-555.
[161] S. K. H. Kim Ju Young, "Methodology for Automatic Synthesis of Wargame Simulator using DEVS," *Advanced Communication Technology, The 9th International Conference on,* vol. 1, pp. 436-441, 2007.
[162] P. T. Center. (2009, 11/10/08). *Simulex, Inc.* Available: http://www.simulexinc.com
[163] K. B. David P. Landau, *A Guide to Monte Carlo Simulations in Statistical Physics*: Cambridge University Press, 2005.
[164] P. H. Frédéric Adam, *Encyclopedia of Decision Making and Decision Support Technologies*: Information Science Reference, 2008.
[165] D. K. L. Ben L. Titzer, Jens Palsberg, "Avrora: scalable sensor network simulation with precise timing," in *Information Processing In Sensor Networks, Proceedings of the 4th international symposium on Information processing in sensor networks*, Los Angeles, California, 2005.

[166] J. Ousterhout, "Tcl - Tool Command Language," 8.5.6 ed, 1988.
[167] J. Ousterhout, "Tk," 8.5 ed, 1991.
[168] DARPA. (2009, 04/12/09). *DARPA | Home*. Available: http://www.darpa.mil/
[169] PARC. (2002-2009, 04/12/09). *Palo Alto Research Center (PARC)*.
[170] Berkeley. (2009, 04/12/09). *University of California, Berkeley*. Available: http://berkeley.edu/
[171] TG4, "IEEE 802.15.4," 2003.
[172] B. Stroustrup, *The C++ Programming Language*, 1986.
[173] L. The VINT Project. A collaboratoin between researchers at UC Berkeley, USC/ISI, and Xerox PARC. Kevin Fall, Kannan Varadhan, , "The ns Manual," 2005.
[174] J. R. J. Postel, "FILE TRANSFER PROTOCOL (FTP) - RFC 959," ed. Network Working Group - Request for Comments, 1985.
[175] B. Shneiderman, *Designing the user interface: strategies for effective human-computer interaction*: Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1997.
[176] J. Preece*, et al.*, "Interaction design: beyond human-computer interaction," 2002.
[177] I. MacKenzie, "Fitts Law as a Research and Design Tool in Human-Computer Interaction," *Human-Computer Interaction,* vol. 7, pp. 91-139, 1992.
[178] J. Foley*, et al.*, "The human factors of computer graphics interaction techniques," 1990, pp. 67-121.
[179] J. F. Alan Dix, Gregory Bowd, RussellBeale, *Human-Computer Interaction*, 3rd ed.: Pearson Education Limited, 2004.
[180] P. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement.," *Journal of Experimental Psychology,* vol. 47, pp. 381 - 391, 1954.
[181] T. W. Malone, *Heuristics for designing enjoyable user interfaces: lessons from computer games*. Norwood, NJ, USA: Ablex Publishing Corp, 1984.
[182] *Java Servlet Technology - Implementations & Specifications,* S. D. N. (SDN), 1994-2009.
[183] Apache Software Foundation, "Apache Tomcat," Apache, Ed., 6.0.16 ed, 1999.
[184] Sun, "Java SE Desktop Technologies - Java Beans -," ed, 2000.
[185] W3C. (1994, 03/05/10). *World Wide Web Consortium (W3C)*. Available: http://www.w3.org/
[186] Macromedia, "FutureSplash Animator," ed, 1996.
[187] Macromedia, "Macromedia Flash MX 2004," ed, 2003.
[188] Netscape Mozilla, "JavaScript," 1.8 ed, 1995.
[189] Ecma International, "ECMA-262 ECMAScript," vol. 262, ed, 1997.
[190] T. Reenskaug, "MODELS - VIEWS - CONTROLLERS," Xerox PARC10/12/79 1979.
[191] A. Kay, "Smalltalk-72 instruction manual. Xerox PARC Rep," SSL-76-6, Xerox Palo Alto Research Center, Palo Alto, Calif1972.
[192] J. Coutaz, "PAC: AN OBJECT ORIENTED MODEL FOR IMPLEMENTING USER INTERFACES," *SIGCHI Bull.,* vol. 19, pp. 37-41, 1987.
[193] J. H. K Brayer, "Evaluation of error detection polynomial performance on the AUTOVON channel," in *National Telecommunications Conference*, New Orleans, La, USA, 1975, pp. 8 - 21.
[194] W. Commons, "Exposed terminal problem," ed: Wikipedia, 2006.
[195] Microsoft, "DirectX," 9.0c ed, 2004.
[196] Silicon Graphics, "OpenGL," 3.2 ed, 2009.
[197] Valve, "Counter-Strike: Source," ed: Steam, 2005.
[198] Valve. (2006, 1/12/09). *Source Multiplayer Networking*. Available: http://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking
[199] id Software, "Quake," 1.09 ed, 1996.
[200] Telekinesys Research Ltd, "havok," 6.0 ed, 2008.
[201] Blizzard Entertainment, "World of Warcraft," ed: Vivendi Universal, 2005, p. MMORPG.
[202] V. Software, "Counter-Strike," 1.6 ed: Vivendi Universal, 2003, pp. First person shooter as a modification to Half-Life.
[203] Valve, "Hammer," 4.1 ed: Valve, 2008.
[204] id Software, "GtkRadiant," 1.5.0 ed, 2007.
[205] Blizzard Entertainment, "Warcraft III World Editor," 1.22a ed, 2008.
[206] A. Burr, "Moo Mapper," 0.94 ed, 2005, p. Level editor.
[207] Aion Team Development Department, "Aion," 1.5 ed: NCSoft, 2008.
[208] CCP Games, "EVE Online," ed: SSI, CCP Games, 2003, p. MMORPG.

[209] Linden Labs. (2009, *Second Life | LindeX; Market Data*. Available: http://secondlife.com/statistics/economy-market.php
[210] A. Marios and T. Velin, *A distributed architecture for MMORPG*. Singapore: ACM, 2006.
[211] Active Worlds, "Active Worlds," 4.1 ed, 1995.
[212] Indiana University Center for Research on Learning and Technology, "Quest Atlantis," ed, 2001.
[213] X. Hui, "HiPiHi," 50022 ed. Beijing: HiPiHi Co. Ltd, 2005.
[214] X. Hui. (2007, 24/06/09). *HiPiHi™ – China's Pioneer of The 3D Virtual World*. Available: http://www.hipihi.com/aboutus_en.html
[215] G. Lu. (2007, 24/06/09). *HiPiHi - A Virtual World Born in China*. Available: http://www.readwriteweb.com/archives/hipihi_china_virtual_world.php
[216] M. O. Adam Frisby, Stefan Andersson, Sean Dague, Brian McBee et al. (2007, 01/08/10). *OpenSimulator*. Available: http://opensimulator.org/
[217] Second Life Reverse Engineering Team, "libsecondlife," ed, 2007.
[218] C. Consortium. (2008, 10/02/09). *Croquet*. Available: http://croquetconsortium.org/index.php/Main_Page
[219] J. L. Alan Kay, Mark McCahill, Andreas Raab, "Croquet Software Developer's Kit," ed, 2007.
[220] D. I. Alan Kay, Adele Goldberg, XEROX Parc, "Squeak," ed, 1996.
[221] Linden Research Incorporated, "Linden Scripting Language," ed, 2003.
[222] C. Linden. (2009, 18/11/09). *Second Life Blogs: Community: Our Content Management Roadmap*. Available: https://blogs.secondlife.com/community/community/blog/2009/08/04/our-content-management-roadmap
[223] Linden Labs. (2007, 11/02/09). *LSL Portal - Second Life Wiki*. Available: http://wiki.secondlife.com/wiki/LSL_Portal
[224] CatherineOmega. (2008, 01/10/08). *LSL Wiki*. Available: http://www.lslwiki.net/
[225] *STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES,* I. E. T. F. (IETF), 1982.
[226] S. A. Phillip Merrick, Joseph Lapp, "XML remote procedure call (XML-RPC)," U.S.A Patent 7028312, Apr 11, 2006, 1999.
[227] *Hypertext Transfer Protocol -- HTTP/1.1,* N. W. Group 2068, 1997.
[228] Linden Research Inc. (2009, 12/10/09). *LlEmail - Second Life Wiki*. Available: http://wiki.secondlife.com/wiki/LlEmail
[229] Linden Research Inc. (2009, 12/10/09). *LlOpenRemoteDataChannel - Second Life Wiki*. Available: http://wiki.secondlife.com/wiki/LlOpenRemoteDataChannel
[230] Linden Research Inc. (2009, 06/09/09). *LlSendRemoteData - Second Life Wiki*. Available: http://wiki.secondlife.com/wiki/LlSendRemoteData
[231] Linden Research Inc. (2009, 20/03/10). *LlHTTPRequest - Second Life Wiki*. Available: http://wiki.secondlife.com/wiki/LlHTTPRequest
[232] Linden Research Inc. (2009, 26/05/09). *HTTP Texture - Second Life Wiki*. Available: http://wiki.secondlife.com/wiki/HTTP_Texture
[233] S. Colebourne, "Joda Time," 1.0 ed, 2005.
[234] Apache Software Foundation. (2009, 12/02/06). *Apache Commons*. Available: http://commons.apache.org/
[235] Red Hat, "Hibernate," 3.3.2 ed, 2009.
[236] The CVS Team, "Concurrent Versions System," 1.11.23 ed, 1990.
[237] CollabNet, "Subversion," 1.6.6 ed, 2000.
[238] J. H. Linus Torvalds, "git," 1.6.5.2 ed, 2009.
[239] H. Van Vliet and J. Van Vliet, "Software engineering: principles and practice," 2000.
[240] I. Sommerville, *Software Engineering*, 6 ed.: Addison Wesley, 2000.
[241] DrQueue.org, "DrQueue, the Open Source Distributed Render Queue," 0.64.3 ed, 2009.
[242] U. o. S. A. School of Computer Science. (2007, 12/02/08). *WiFi Virtual Laboratory*. Available: http://wifi.cs.st-andrews.ac.uk/
[243] *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4,* N. W. Group RFC 4728, 2007.
[244] *Ad hoc On-Demand Distance Vector (AODV) Routing,* N. W. Group RFC 3561, 2003.
[245] *Temporally-Ordered Routing Algorithm (TORA),* N. Group, 2001.

[246] University of Hawaii, "ALOHAnet," ed, 1970.
[247] Sun, "Java Servlet Technology," ed, 1997.
[248] W. W. W. C. (W3C), "XHTML™ 1.0 The Extensible HyperText Markup Language," in *A Reformulation of HTML 4 in XML 1.0* vol. 1.0, ed. http://www.w3.org/TR/2000/REC-xhtml1-20000126/: W3C, 2000.
[249] Joint Photographic Experts Group. (1992, *JPEG*. Available: http://www.jpeg.org
[250] E. G. Kent Beck, David Saff, "JUnit," 4.5 ed, 2008.
[251] *Transmission Control Protocol - DARPA Internet Program Protocol Specification,* RFC 793, 1981.
[252] *User Datagram Protocol,* RFC RFC 768, 1980.
[253] L. The VINT Project. A collaboratoin between researchers at UC Berkeley, USC/ISI, and Xerox PARC. Kevin Fall, Kannan Varadhan,. (2009, 12/02/09). *The ns Manual (formerly ns Notes and Documentation)*. Available: http://www.isi.edu/nsnam/ns/doc/index.html
[254] M. A. Jean-Loup Gailly, "gzip," 1.4 ed, 1992.
[255] Adobe, "ActionScript," 3.0 ed, 1998.
[256] K. Rose. (2004, 12/11/06). *Digg - The Latest News Headlines, Videos and Images*. Available: http://digg.com/
[257] J. Brooke, *SUS: a "quick and dirty" usability scale, in Usability Evaluation in Industry.*, 1 ed.: CRC Press, 1996.
[258] C. Allison, Miller, A, Getchell, K, Sturgeon, T., "Exploratory Learning for Computer Networking," in *Advances in Web Based Learning – ICWL 2007*. vol. 4823, ed: Springer Berlin / Heidelberg, 2008, pp. 331-342.
[259] Adobe Systems, "Adobe Flex," 3.3.0 ed, 2004.
[260] Open-source, "Eclipse ", 3.4.2 ed, 2004.
[261] Adobe, "MXML 2009 - Functional and Design Specification," ed, 2009.
[262] Microsoft, "Microsoft OFfice Visio Professional 2003," 11.3216.6360 ed, 2003.
[263] J. Malek, "Trace graph - Network Simulator NS-2 trace files analyser," ed, 2007.
[264] The Mathworks Inc., "MATLAB® - The Language of Technical Computing," ed, 1994-2008.
[265] D. G. a. T. Morgner, "JFreeChart," ed, 2005.
[266] F. F. H. Nah, "A study on tolerable waiting time: how long are Web users willing to wait? ," *Behaviour & Information Technology,* vol. 23, pp. 153-163, 2004.
[267] D. M. Mark Zuckerberg. (2009, 11/12/08). *Welcome to Facebook!* Available: http://www.facebook.com
[268] E. W. Jack Dorsey, Biz Stone. (2006, 01/11/08). *Twitter: what are you doing?* Available: http://twitter.com/
[269] Linden Research Incorporated. (2003, *Second Life*. Available: http://secondlife.com/
[270] L. R. Incorporated. (2008, *Second Life*. Available: http://secondlife.com/
[271] Linden Labs. (2009, 29/09/09). *Economic Statistics (Raw Data Files)*. Available: http://secondlife.com/statistics/economy-data.php
[272] L. Grimmer. (2006, 22/11/08). *Interview with Ian Wilkes from Linden Lab*. Available: http://dev.mysql.com/tech-resources/interviews/ian-wilkes-linden-lab.html
[273] Debian Project, "debian," ed, 1993, p. Open source software.
[274] E. Rosebrock, *Setting Up LAMP: Getting Linux, Apache, MySQL, and PHP Working Together* John Wiley & Sons, 2004.
[275] Linden Labs, "Title," unpublished.
[276] A. Semuels. (2008, 01-02-08). *Virtual world is make-believe, but crooks are real*. Available: http://www.baltimoresun.com/technology/bal-bz.secondlife23jan23,0,4984393.story
[277] Linden Research Inc. (2005, 18/11/09). *Teen Second Life*. Available: http://teen.secondlife.com/
[278] Z. Barkley. (2007, 12/02/09). *Pear Computer/Laptop - Open Source!* Available: https://www.xstreetsl.com/modules.php?name=Marketplace&file=item&ItemID=713374
[279] Linden Labs. (2009, 11/03/09). *Rez - Second Life Wiki*. Available: http://wiki.secondlife.com/wiki/Rez
[280] Linden Labs. (2009, 16/03/09). *LlParticleSystem - Second Life Wiki*. Available: http://wiki.secondlife.com/wiki/LlParticleSystem
[281] Linden Labs. (2008, 12/02/09). *Category:LSL Notecard - Second Life Wiki* [Web Page]. Available: http://wiki.secondlife.com/wiki/Category:LSL_Notecard

[282] A. Marcus, "Return on investment for usable UI design," *User Experience,* vol. 1, pp. 25-31, 2002.
[283] C. Lewis and J. Rieman, *Task-centered user interface design*, 1993.
[284] R. Virzi*, et al.*, "A comparison of three usability evaluation methods: Heuristic, think-aloud, and performance testing," 1993, pp. 309-313.
[285] N. Denzin and Y. Lincoln, "The Handbook of Qualitative Research 3ed."
[286] R. Kumar, *Research Methodology*: S.B. Nangia for APH Publishing Corporation, 2008.
[287] Linden Research Inc. (2009, 10/07/09). *Mega Prim - Second Life Wiki*. Available: http://wiki.secondlife.com/wiki/Mega_Prim
[288] J. Kinicki and M. Claypool, "Traffic analysis of avatars in Second Life," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2008, pp. 69-74.
[289] Linden Research Inc. (2009, 10/07/09). *LSL Alternate Editors* [Web Page]. Available: http://wiki.secondlife.com/wiki/LSL_Alternate_Editors
[290] M. N. Vlad Bjornson, Sonja Manatiso, Hippi Hammerer, Baron Hauptmann, "LSL-Editor," 2.39 ed, 2008.
[291] OpenSimulator. (2010, 02/03/10). *OpenSim Archives - OpenSim*. Available: http://opensimulator.org/wiki/OpenSim_Archives
[292] Metaverse. (2009, 11/06/09). *Meerkat (0.2.2 (211) ed.)*. Available: http://www.meerkatviewer.org
[293] OpenSimulator. (2008, 01/03/10). *Talk:OSSL Proposals - OpenSim*. Available: http://opensimulator.org/wiki/Talk:OSSL_Proposals
[294] OpenVCE. (2009, 02/03/10). *OpenVCE | ...Open Virtual Collaboration Environment*. Available: http://openvce.net/
[295] E. H. Modiano, "6.263J Data Communication Networks, Fall 2002. (Massachusetts Institute of Technology: MIT OpenCourseWare)," ed, 2002.
[296] L. Zheng, "6.452 Principles of Wireless Communications, Spring 2006. (Massachusetts Institute of Technology: MIT OpenCourseWare)," ed, 2006.
[297] e. a. Markey, "Secret Communication System," United States of America Patent, August 1942, 1941.
[298] C. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal,* vol. 27, pp. 379 - 423, 623-656, 1948.
[299] A. Inc., "Darwin," 3 ed, 2000.
[300] IETF, "Real Time Streaming Protocol (RTSP)," vol. 2326, ed, 1998, p. 92.
[301] R. Gallager, "Gallager, Robert, 6.450 Principles of Digital Communications I, Fall 2006. (Massachusetts Institute of Technology: MIT OpenCourseWare)," ed, 2006.
[302] OpenSimulator. (2009, 02/03/10). *0004435: Delete many prims using llDie() at once, some "sort of" don't go away* Available: http://opensimulator.org/mantis/view.php?id=4435
[303] P. Steenkiste, "The use of a controlled wireless testbed in courses," presented at the Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education, Paris, France, 2009.
[304] Novell, "Mono," 1.2.6 ed, 2007, p. Open source implementation of Microsofts .NET platform.
[305] Linden Research Inc., "Second Life Viewer," 1.23.4 ed, 2009.