Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2015) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 10th International Conference on Future Networks and Communications (FNC 2015)

HTTP/2 and QUIC for Virtual Worlds and the 3D Web?

Hussein Bakri[a], Colin Allison[a]*, Alan Miller[a], Iain Oliver[a]

[a]*School of Computer Science, University of St Andrews, St Andrews KY16 9SX, United Kingdom*

Abstract

The continuing advances in computer graphics and Internet bandwidths are supporting a gradual convergence between multi-user virtual worlds (MUVW), such as Second Life and OpenSim (SL/OS), and the nascent 3D Web. However, significant networking barriers remain to exploiting these capabilities for developing the 3D Web. These barriers include latency of content update and firewall blocking. In MUVWs the firewall and latency problems are related as the SL/OS network protocols designed over twelve years ago sought to minimise latency through the use of multiple concurrent UDP-based virtual circuits. Most firewall administrators are loathe to open up over fifty unknown UDP ports to accommodate such applications. New protocols now being deployed on the web such as SPDY, HTTP/2 and QUIC seek to reduce latency and routinely traverse firewalls. One of the key goals of the convergence between MUVWs and the 3D Web is for MUVW functionality to be provided in a standard web browser, with optional links to other autonomous virtual worlds. It follows that as an incremental step towards the 3D Web the use of these new web protocols in MUVWs should be researched. This paper details traffic management approaches in SL/OS MUVWs, clarifies 3D Web concepts and terminology, explains the functionality provided by the new web protocols, and provides a mapping which postulates how their features can be exploited for the benefit of MUVWs as part of the convergence with the 3D Web.

Keywords: HTTP/2, SPDY, QUIC, Multi-User Virtual World, Web-Based Virtual World

1. Introduction

Multi-User Virtual Worlds (MUVWs) are a type of Internet application where users interact with an immersive 3D environment and with each other through *avatars*. Unlike Massive Multiplayer Online Games, MUVWs have no particular goals and their platforms can be seen as prototypes for the 3D Web. Part of our motivation for using Virtual Worlds is that students readily engage with them for educational purposes, often more so than with

* Corresponding author: e*mail address:* ca@st-andrews.ac.uk

conventional learning materials and contexts [1] [2]. Virtual Worlds for education have been created to support topics including 802.11 WiFi [3], Internet Routing (see Fig. 1) [4] and programming algorithms [2] [5].

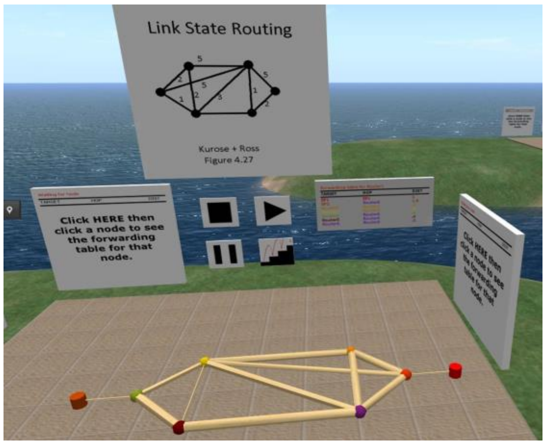
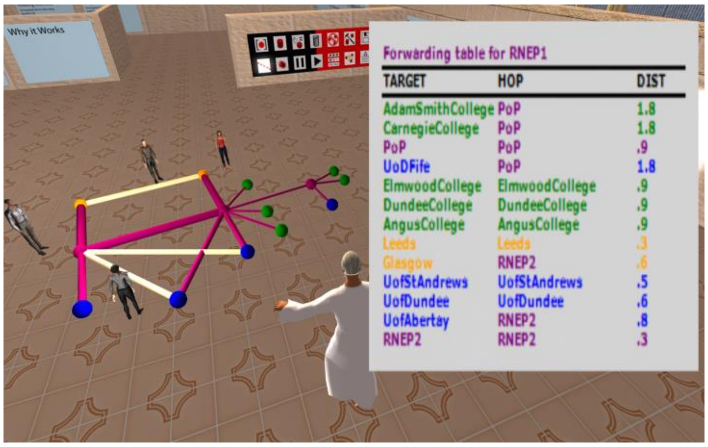

Link State Routing
Kurose + Ross
Figure 4.27
Click HERE then click a node to see the forwarding table for that node.
Click HERE then click a node to see the forwarding table for that node.

Figure 1a: Interactive example from a popular networking text[†] in an immersive learning enviromnent



Forwarding table for RNEP1

TARGET	HOP	DIST
AdamSmithCollege	PoP	1.8
CarnegieCollege	PoP	1.8
PoP	PoP	.9
UoDFife	PoP	1.8
ElmwoodCollege	ElmwoodCollege	.9
DundeeCollege	DundeeCollege	.9
AngusCollege	AngusCollege	.9
Leeds	Leeds	.3
Glasgow	RNEP2	.6
UofStAndrews	UofStAndrews	.5
UofDundee	UofDundee	.6
UofAbertay	RNEP2	.8
RNEP2	RNEP2	.3

Figure 1b: A group of students cooperatively model their regional network and see the effects of the routing algorithm on a broken link

Second Life [6] was pioneering in its global reach in 2003 but it was not designed for educational exploration and several commentators have highlighted problem areas that arise when using it for that purpose [7][8]. These include: commercial cost, code size restrictions, lack of an integrated development environment, difficulty of coursework marking due to the ownership and permissions system, poor quality of experience due to poor scalability in terms of avatars per region and firewall blocking by campus computing services. Open Simulator (OpenSim) [9] has increasingly displaced Second Life (SL) as the platform of choice for developing immersive learning environments. OpenSim is an open source project which mostly uses the same protocols as Second Life so works with most SL compatible viewers/clients. This compatibility has resulted in OpenSim becoming a de facto standard and platform for a wide range of Virtual Worlds. In addition, OpenSim introduced the idea of a Hypergrid [10] whereby links, which can be followed by avatars, can be established between independent and autonomous Virtual Worlds. Thus, OpenSim with its Hypergrid can be seen as a prototype for the 3D Web.

While OpenSim offers solutions to many of the drawbacks encountered with Second Life there are still features (or the lack thereof) inherited from Second Life which act as barriers to further developing OpenSim and the Hypergrid as a prototype 3D Web. In particular firewall blocking at the edge of campus networks stops the routine sharing of resources and establishment of links [11]. In addition, Virtual Worlds need to more efficiently utilise the network resources available. For example, bandwidth management is not linked to key user Quality of Experience (QoE) parameters established as benchmarks in [12] such as frames per second or frame time, and poor Quality of Service (QoS) can in turn lead to poor QoE. A further consideration is the extent to which Virtual World traffic is a good network citizen, not hogging limited resources during transient peaks at the expense of traffic that is better self-regulated.

This paper examines two relatively recent web communication protocols – HTTP/2 and QUIC - with a view to assessing their suitability as alternatives to the current Second Life / OpenSim (SL/OS) communication regime. Can they solve problems of reachability associated with firewall blocking and inadequate QoS due to traffic management policies? The next section reviews the SL/OS protocols and the underlying requirements they seek to address in terms of real time multimedia traffic. Section 3 outlines the concepts and technologies used by the nascent 3D Web. The fourth section explains HTTP/2 and the fifth section briefly summarizes criticisms of it; section six describes

[†] Kurose, J. and K. Ross, *Computer Networking: A Top-Down Approach, 6th ed.* 2012: Pearson. 896p.

QUIC and the seventh section postulates how features from these two new web protocols could profitably be used for MUVWs and an OpenSim-based prototype 3D Web. Section 8 summarizes and concludes.

2. MUVWs Traffic Management

SL/OS has a client/server model where the simulation of the environment is executed on the server [13][14][15]. The SL/OS application-level protocol mostly (approx. 98%) uses UDP [16][17]; TCP is only used for user log-on and authentication [18][19]. There are more than 473 different types of SL/OS UDP messages [20]. SL/OS protocols are described in depth in [21]. Server components are described in [22].

Several studies have measured SL/OS traffic. Some focus on what is happening on the client side in terms of different concentrations of objects and avatars for defined avatar activities [13][15]. Other studies have developed crawlers to exploit server side statistics on avatar locations and capabilities and the evolution of objects in regions [23][24]. Creating models of SL/OS traffic is investigated in [14]. *Textures* and *Primitives* (Prims) constitute the majority of SL/OS traffic, ranging from hundreds of megabytes to gigabytes being served from the server to the clients [25].

SL/OS implements adaptive traffic management mechanisms which involve a throttle mechanism to limit the maximum rate of the downloading bandwidth to a client. This mechanism is usually aggressive and in some cases fails to reduce the transmission rate in line with a TCP fair algorithm in the presence of loss and delay which can lead to a waste in available bandwidth and contribute towards network congestion. MUVWs traffic has soft real time requirements and is sensitive to delay, jitter and bandwidth constraints. Normally the throttle value is calculated as 1.5 times the value set by the user in his/her viewer. Making virtual worlds clients and servers behave in a TCP fair way was addressed by the experimental Mongoose Client and Server [26][27]. The throttling mechanism and detailed analysis of SL/OS traffic can be found in [16][26][17][27].

SL/OS divides communications into UDP-based circuits [16] which are allocated the following proportions of the bandwidth: Asset (16%) for transmitting and receiving information about avatar inventory; Texture (27%) for images applied to primitives; Task (27%) for packets relating to primitives, the particle system, trees and avatar control traffic; Wind (2.8%) for simulating wind behavior by moving trees and making the sound of wind; Land (14%) for land and water height maps; Cloud (2.8%) for the distribution and movement of clouds; and Resend (10% of bandwidth) for “reliable packets” resent from other circuits with no ACKs [28]. Circuits can provide reliability for some of the packets whilst others remain unreliable. Circuits use sequence numbers and timeouts to detect packet loss.

The SL/OS traffic management mechanism tracks round trip time (RTT) and congestion level, performs application level framing and provides some reliability for some types of packet. RTT is determined by using special ping packets that the server and client respond to. This determines the amount of time to wait for missing packets before they are considered lost. It is also used to determine the length of time to wait for reliable packets to be acknowledged before resending them on the Resend circuit. The downloading bandwidth to SL/OS viewers is governed by the throttle system. The highest level of bandwidth utilization is when the avatar is flying or teleporting [19].

Certain circuits and certain type of MUVW traffic needs to be prioritized on occasion. For example, Avatar control. Lack of prioritization of certain circuits especially under harsh measurements taken by the throttling mechanism, leads to users experiencing problems such as their avatars appearing as white clouds, islands appearing to be empty of objects as these information are not sent yet or being demoted by other traffic. A common problem encountered by users is that their avatars react slowly to commands especially in the beginning of a session. Errors in avatar positions (avatars seems to be halfway through walls), objects showing before land and many other odd effects are in the majority of cases due to poor network traffic management; some data is being received by the client before other more important data [17]. Better bandwidth allocation to each circuit results in less delay and more accurate avatar and object representations leading to better Quality of Experience and fairness to competing traffic.

3. The 3D Web

In this section, we present a brief overview of what the “3D web” means and give some examples of how MUVWs are converging with the 3D Web. The “3D Web” is a general term encompassing several technologies. It can mean to navigate the web in 3D; to embed “Web3D” technologies into the 2D Web; to refer to the idea of Hyper-Grids or Internet of Metaverses where an avatar can move between virtual environments of different types. Web3D technologies such as X3D, O3D, Unity3D, WebGL and others have facilitated the building of Multi-User 3D Virtual Worlds on the web.

“Web-Based Virtual Viewers” (WBVVs) for MUVWs are alternatives to the stand-alone viewers used in traditional SL/OS MUVWs such as the Second Life viewer, Hippo and Phoenix. WBVVs work in any standard web-browser. One of the benefits of WBVV is that it removes the need to download and install stand-alone software package. Even a plugin represents a considerable reduction in efforts. These viewers can leverage Web3D technologies such as WebGL which is now a standard part of the major browsers. For example, the Unity3D plugin based Virtual Viewer that connects and renders SL/OS scenes [29]. Another example is the Pixie Viewer [30] which runs in any HTML5/WebGL browser and can be connected to a standalone server or to OpenSim grids (needs an extra module in the OS server).

“Web-Based Virtual Worlds” (WBVWs) are similar to traditional MUVWs like SL/OS but appear to be completely integrated into the web from the perspective of the user. All the user has to do is to navigate the environment from any web browser. WBVWs are also built using Web3D tools and languages. One example of WBVW is the Virtual World Framework (VWF) [31][32]. VWF is a means to connect virtual worlds and 3D entities and content via web browsers. It is an open source platform that allows anyone to build collaborative 3D applications on the web. Sandbox [33] is a Virtual World Framework (VWF) authoring and delivery platform for creating 3D environments on the web. Users can create complex and beautiful virtual worlds on the web. The tool uses WebGL so does not require a plug-in. Another example is the ReactionGrid Jibe Platform [34].

The web is currently moving over to the latest version of its standard application protocol: HTTP/2. Google are experimenting with another web protocol called QUIC. Both of these web protocols are explained in the next section. We then go on to posit that HTTP/2 and QUIC are highly suitable as alternative protocols for virtual worlds whether Web-Based Virtual Worlds or Multi-User Virtual worlds like SL/OS.

4. HTTP/2

HTTP/2 [35] [36] [37] is a major enhancement to HTTP, principally motivated by the desire to reduce the Page Load Time (PLT) of modern, large, complex web pages. The IETF HTTPbis working group published internet draft standard 17 [35] in early 2015; the protocol is in its final stages before becoming an official standard [39]. Table 1 summarises the evolution of HTTP from HTTP 1.0 to HTTP/2.

HTTP 1.0	HTTP 1.1	HTTP/2
Sequential ASCII request-response messages over TCP	Sequential ASCII request-response messages over TCP	Full duplex binary frames over TLS/TCP or TCP
PDU: HTTP Message	PDU: HTTP Message	PDU: HTTP/2 Frame (10 Types)
New TCP connection for each Request/Response Browsers open multiple parallel TCP connections within same domain	Persistent TCP connection Browsers open multiple parallel TCP connections within same domain Pipelining specified but not mandatory and not implemented.	One persistent TCP Connection per domain Pipelining mandatory Stream Multiplexing Dynamic stream dependencies (Re)Prioritization of streams
Caching, Content compression option	Caching, content compression option	Caching, Content compression Header Compression Server Push Flow Control

Table 1: HTTP 1.0, 1.1 and HTTP/2 overview of main features

HTTP/2 was initially a copy of Google's SPDY, and SPDY and HTTP/2 are used interchangeably in many papers due to the great influence of SPDY on HTTP/2.

Comparisons of the gains in PLT by increasing bandwidth vs reducing aggregate RTT showed that the latter was clearly more effective [38] so HTTP/2 seeks to reduce the number of RTTs. HTTP/2 also seeks to reduce the number of concurrent TCP connections from a browser to the same domain. Although HTTP 1.1 added pipelining to HTTP 1.0, the feature was rarely implemented due to interference from proxies and *Head of Line Blocking* whereby a single large or time-consuming object request blocks all the others behind it in the pipeline. HTTP/2 addresses this by introducing concurrency, prioritization and multiplexing of requests. With regard to security, SPDY always uses TLS (TCP port 443), but HTTP/2 also allows unencrypted traffic (TCP port 80). The following subsections give some more detail of the new features found in SPDY and HTTP/2.

4.1. Frames and Streams; Request and Response Multiplexing

The unit of communication in HTTP/2 is the frame. There are ten different types: DATA, HEADERS, PRIORITY, RST_STEAM, SETTINGS, PUSH_PROMISE, PING, GOAWAY, WINDOW_UPDATE, CONTINUATION. A stream in HTTP/2 consists of bidirectional sequences of frames flowing between two endpoints (client and server). In other words, the server and client can send data simultaneously. Multiplexing allows for any number of bidirectional streams of request and response frames (of maybe similar or different data) on a single TCP connection.

4.2. Prioritization, Reprioritization and Dependency of Streams

Streams can be interleaved and prioritized. This is to allow an endpoint to allocate more resources to what is being prioritized when managing concurrent streams. A client can assign a priority number for a new stream in the HEADERS frame. Reprioritization of streams can be regulated by the PRIORITY frames.

Streams can explicitly depend on the completion of other streams. This also affects the priority of streams. Dependency is assigned a weight between 1 and 256 inclusive. Dependent streams share the resources assigned to their parent in accordance with the weight assigned to them. Dependent streams move with their parent stream whenever the parent is reprioritized. A stream that is not dependent on any other stream is given a weight of 0 [35].

4.2.1. Binary Framing Layer

The binary framing layer in SPDY "dictates how the HTTP messages are encapsulated and transferred between the client and server" [40]. HTTP/2 has kept the same semantics, such as verbs and headers of HTTP 1.x. Changes occurred on the level of how theses semantics were encoded, encapsulated and then transferred.

4.3. Server Push

A server can send pre-emptively (or "push") additional information in addition to replying to requests from clients. For example, a server can send images, icons, CSS or JavaScript code before the client explicitly requests them. A client can request that server push be disabled during a connection. A server sends the response (pushed data) and the actual request that the client would have to send in order to receive the response in a specific frame called PUSH_PROMISE.

4.4. Header Compression

In HTTP 1.x, headers can be repetitive and verbose. HTTP/2 compresses headers using the HPACK algorithm [42], based on Huffman encoding.

4.5. Flow Control

HTTP/2 allows the use of flow control algorithms for both individual streams and for the connection as a whole. This supports better utilization of network resources by not allowing a particular stream to starve, and by dealing with slow/fast upstream and downstream connections adequately. It is a hop by hop directional credit-based scheme. WINDOW_UPDATE frames advertise how many octets can be received for a specific stream or for the whole connection. The sender must respect flow control limits advertised by the receiver. Only DATA frames are subject to its effect.

4.6. RTT and Liveness

PING frames have the highest priority. They are used to measure round trip time and check if the connection is still functional or the peer is still alive.

4.7. SPDY and HTTP/2 performance

Studies have compared SPDY to previous HTTP versions [43] [44]. These give mixed, sometime contradictory, results in term of SPDY outperforming older versions of HTTP or the opposite. Part of the problem for SPDY (and HTTP/2) not outperforming HTTP 1.x lies in the behavior of TCP (see [43] [45]). A single TCP congestion avoidance window can put SPDY or HTTP/2 at a disadvantage compared with many TCP connections each with a separate congestion window, which is often the case with HTTP 1.x versions. In addition, a single packet lost can stall (or at least impact) all the multiplexed streams on a single TCP connection. SPDY or HTTP/2 was studied also on mobile devices [47][45] and high latency Satellite networks [48][49] .

5. Criticisms of HTTP/2

HTTP/2 started as a copy of SPDY in 2012, and was almost fully ratified as an Internet standard by early 2015. Criticisms of the protocol and the nature of its standardization are summarized in [39]. These include technical issues such as: too much needless complexity; protocol layering violations by replicating TCP functionality: both protocols support flow control, window size negotiation and pipelining; SPDY introduced explicit state to HTTP, by way of session initiation and closedown, in a similar way to the TCP macro state. In addition most implementations have adopted the SPDY approach of all traffic being run over TLS, regardless of the cost or need. Khalid et al. [41] have argued that Sever Push can be problematic in mobile devices because it can waste battery or bandwidth and proposes mechanisms to be adopted in HTTP/2 that adjust the overall performance on mobile devices. Standardization process issues include: too much rush; not enough community consultation; too much influence from commercial interests; missed opportunities such as improved privacy e.g. why not get rid of cookies?

6. The QUIC Protocol

QUIC [50][51][52][53] stands for “Quick UDP Internet Connections”. It is an experimental web protocol from Google that is an extension of the research evident in SPDY and HTTP/2. Table 2 gives an overview comparison with HTTP/2.

QUIC	**HTTP/2**
Runs over UDP	Runs over TCP (ports 80, 443)
Multiplexing multiple requests/responses over one UDP pseudo-connection per domain	Multiplexing multiple requests/responses over one TCP connection per domain
Promises to solve Head Of Line Blocking at the Transport Layer (caused by TCP behaviour)	Promises to solve Head Of Line Blocking at the Application layer (caused by HTTP 1.1 pipelining)
Best case scenario (in repeat connections, client can send data immediately (Zero Round Trips)	Best Case Scenario (1 to 3 Round Trips for TCP connection establishment and/or TLS connection)
Reduction in RT gained by features of the protocol such as Multiplexing over one connection etc…	Reduction in RTs in comparison to HTTP 1.X gained by features such as Multiplexing over one connection, and Server Push
HTTP/2 or SPDY can layer on top of QUIC (i.e. all features of SPDY are supported in QUIC)	HTTP/2 or SPDY can layer on top of QUIC or TCP
Packet-level Forward Error Correction	TCP selective reject ARQ used for error correction
Connection migration feature	N/A
Security in QUIC is TLS-like but with a more efficient handshake	Security provided by underlying TLS
TCP Cubic-based congestion control	Congestion control provided by underlying TCP

Table 2: Overview of QUIC compared with HTTP/2

QUIC is premised on the belief that SPDY performance problems are mainly TCP problems and that it is infeasible to update TCP due to its pervasive nature. QUIC sidesteps those problems by operating over UDP instead. QUIC is implemented in the Chrome browser and recognized by recent versions of Wireshark [54]. QUIC is being tested across Google web services from YouTube to Gmail. Although QUIC works on UDP ports 80 and 443 it has (surprisingly) not encountered any firewall problems. QUIC is a multiplexing protocol for exchanging requests and responses over the Internet with lower latency and faster recovery from errors than HTTP/2 over TLS/TCP. QUIC contains some features not present in SPDY such as roaming between different types of networks.

QUIC provides connection establishment with zero round trip time overhead. It promises also to remove Head of Line Blocking on multiplexed streams. In SPDY/HTTP2.0, if a packet is lost in one stream, the whole set of streams is delayed due to the underlying TCP behavior; no stream on the TCP connection can progress until the lost packet is retransmitted. In QUIC if a single packet is lost only one stream is affected [50][51][52][53]. A recent study on the performance of QUIC [55] shows an improved reduction in Page Load Time in comparison with both SPDY and HTTP 1.1. Other studies have focused on security aspects [56] [57].

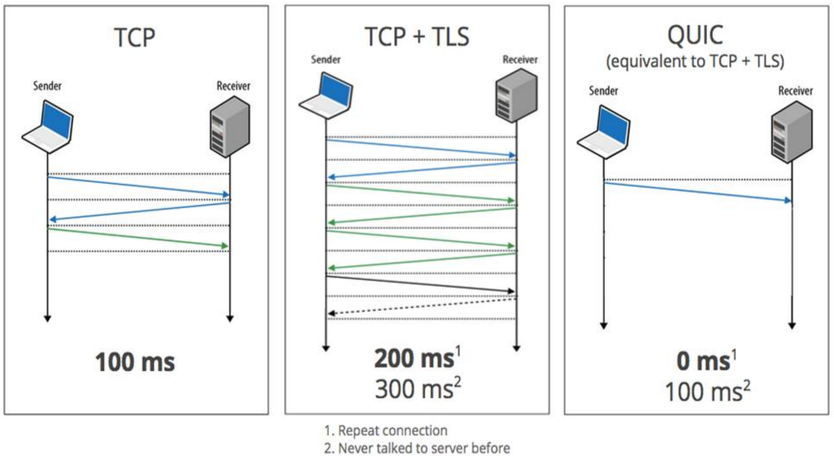


Figure 2: QUIC’s Zero RT handshake

6.1. Multiplexing, Prioritization and Dependency of Streams

QUIC multiplexes multiples streams over a single UDP set of end points [55]. This is of course is not obligatory as it rarely happens on the web due to having several domains. QUIC uses the same prioritization and dependency mechanisms as SPDY [52].

6.2. Congestion control

UDP lacks congestion control so in order to be TCP Fair QUIC has a pluggable congestion control algorithm option. This is currently TCP Cubic.

6.3. Security in QUIC

QUIC provides an ad-hoc encryption protocol named “QUIC Crypto” which is compatible with TLS/SSL. The handshake process is more efficient than TLS. Handshakes in QUIC require zero roundtrips before sending payloads. In TLS on top of TCP this needs between one to three RTTs. QUIC aligns cryptographic block boundaries with packet boundaries. The protocol has protection from IP Spoofing packet re-ordering and Replay attack [57] [58].

6.4. Forward Error Correction (FEC)

A Forward Error Correction mechanism inspired by RAID-4 is available. In the case of one packet being lost in a group, it can be recovered from the FEC packet for the group.

6.5. Connection Migration Feature

QUIC connections are identified by a randomly generated 64 bit CID (Connection Identifier) rather than the traditional 5-tuple of protocol, source address, source port, destination address, destination port. In TCP, whenever a client changes any of these attributes, the connection is no longer valid. In contrast, QUIC has the ability to allow users to roam between different types of connections (for example changing from WiFi to 3G).

7. How HTTP/2 and QUIC can be useful in MUVWs and the 3D Web

This section proposes our ideas about the utility of HTTP/2 and QUIC in MUVWs and WBVWs. We are currently experimenting with possible approaches. HTTP/2 can be used over TLS/TCP or a combination of both protocols by making HTTP/2 layer over QUIC if necessary. SL/OS relies mostly on UDP as the transport protocol so they implement their own traffic control mechanism. By using the same application and transport protocols, Virtual Worlds can become more interoperable with the web facilitating a transition and convergence to the "3D web". Here is how features of the new web protocols can be profitably used in MUVWs and WBVWs:

- **Request and Response multiplexing**: The traffic of SL/OS MUVWs is divided into UDP circuits for Assets, Textures, Task, Wind etc. Multiplexing of streams could be used to multiplex MUVW circuits and can also be used for WBVW traffic.
- **Streams can be interleaved and prioritized**: Certain types of virtual world traffic need to have some precedence over other traffic e.g. avatar control information. This needs a Quality of Service to support a better Quality of Experience. Prioritization of HTTP/2 streams can be used to solve this problem. In addition, f streams can be reprioritized dynamically and thus respond to the current congestion level/or packet loss etc. in each stream. So, for example, if a Texture circuit is congested at a certain point in time, it could be allocated a higher priority. By default, the channel responsible for Avatar control would be prioritized.
- **Server Push**: A server can send additional information in addition to responding solely to requests from clients. This allows a virtual world server to anticipate what a client needs and send virtual region data that might be needed later without the client asking for it. For example, neighboring region information if it is known that the avatar is navigating towards it.
- **Enhanced Security**: HTTP/2 and QUIC use TLS security. This benefits virtual worlds in that it will make their traffic more secure.
- **Flow Control**: HTTP/2 Flow Control allows the use of various flow control algorithms without any changes to the protocol. Flow control is used for both individual streams and for the connection as a whole. It is a hop by hop directional credit based scheme flow control. In addition, QUIC is aiming to be TCP Fair.
- **Traversal of Firewalls**: In most educational settings firewall port blocking inhibits the use of SL/OS as they require a range of "unusual" UDP ports to be opened for effective client-server communication. HTTP/2 has the potential to offer the same functionality through two ports, 80 & 443, which are normally open. QUIC works on these ports with UDP which (surprisingly) does not appear to cause any firewall port blocking.

8. Conclusions and Future Work

We have presented our ideas on how HTTP/2 and QUIC might be utilized in MUVWs and WBVWs. We shed light on current MUVW network protocols and their traffic management with their limitations. We have identified and explained the main concepts and technologies used by the 3D Web and given some examples of Web-Based Virtual Viewers for existing MUVWs and some examples of complete WBVWs. The HTTP/2 and QUIC protocols were explained. The paper shows that theses protocols are good candidates for replacing the current SL/OS network protocols as they can potentially solve problems associated with firewall blocking and traffic management. To the authors' best knowledge, this paper is the first to propose and explain how HTTP/2 and QUIC can be used as underlying protocols for virtual world traffic. As these protocols will work anywhere that HTTP 1.1 does at present they also would be suitable for 3D traffic in Web-Based Virtual Worlds. A prototype that maps HTTP/2/QUIC streams to SL/OS circuits is being developed. Further experiments and traffic measurements are being be done to see if these protocols are as beneficial as we hope.

References

[1] P. Sancho, J. Tirrente, and B. Fernandez-Manjon, “” fighting drop-out rates in engineering education: Empirical research regarding the impact of muves on students’ motivation,” Proc. 1st Eur. Imeersive Educ. Summit, Madrid, 2011.
[2] I. Perera, C. Allison, J. R. Nicoll, and T. Sturgeon, “Towards successful 3D virtual learning - a case study on teaching human computer interaction,” 2009 Int. Conf. Internet Technol. Secur. Trans., 2009.
[3] T. Sturgeon, C. Allison, and A. Miller, “802.11 Wireless Experiments in a Virtual World,” ACM SIGCSE Bull., vol. 41, no. 3, p. 85, Aug. 2009.
[4] J. Mccaffery, A. Miller, and C. Allison, “Extending the use of virtual worlds as an educational platform - Network Island : An Advanced Learning Environment for Teaching Internet Routing Algorithms,” in CSEDU 2011 : Proceedings of the 3rd International Conference on Computer Supported Education, 2011, pp. 279–284.
[5] M. Esteves and B. Fonseca, “Using Second Life for Problem Based Learning in Computer Science Programming,” J. Virtual Worlds Res., vol. 2, no. 1, 2009.
[6] “Second Life Main Website.” [Online]. Available: http://secondlife.com/. [Accessed: 28-Mar-2015].
[7] S. Warburton, “Second Life in higher education: Assessing the potential for and the barriers to deploying virtual worlds in learning and teaching,” Br. J. Educ. Technol., vol. 40, no. 3, pp. 414–426, May 2009.
[8] C. Allison, A. Miller, T. Sturgeon, J. R. Nicoll, and I. Perera, “Educationally enhanced virtual worlds,” Proc. - Front. Educ. Conf. FIE, pp. 1–6, 2010.
[9] “OpenSim Official Website.” [Online]. Available: http://opensimulator.org/wiki/Main_Page. [Accessed: 02-Apr-2015].
[10] “Public Hypergrid Nodes.” [Online]. Available: http://opensimulator.org/wiki/Public_Hypergrid_Nodes. [Accessed: 28-Mar-2015].
[11] C. Allison, A. Campbell, C. J. Davies, L. Dow, S. Kennedy, J. P. McCaffery, A. H. D. Miller, I. A. Oliver, and G. I. U. S. Perera, “Growing the use of Virtual Worlds in education : an OpenSim perspective,” EiED 2012 Proc. 2nd Eur. Immersive Educ. Summit, pp. 1–13, 2012.
[12] A. Sanatinia, I. a Oliver, A. H. D. Miller, and C. Allison, “Virtual machines for virtual worlds,” CLOSER 2012, pp. 108–113, 2012.
[13] S. Fernandes and R. Antonello, “Traffic analysis beyond this world: the case of Second Life,” Proc. Nossdav, Urbana-Champaign, IL, USA, pp. 1–6, 2007.
[14] R. Antonello, S. Fernandes, J. Moreira, P. Cunha, C. Kamienski, and D. Sadok, “Traffic analysis and synthetic models of second life,” Multimed. Syst., vol. 15, no. 1, pp. 33–47, 2009.
[15] J. Kinicki and M. Claypool, “Traffic analysis of avatars in Second Life,” Proc. 18th Int. Work. Netw. Oper. Syst. Support Digit. Audio Video, pp. 69–74, 2008.
[16] I. a. Oliver, A. H. D. Miller, and C. Allison, “Virtual worlds, real traffic,” Proc. first Annu. ACM SIGMM Conf. Multimed. Syst. - MMSys ’10, p. 305, 2010.
[17] I. Oliver, C. Allison, and A. Miller, “Traffic Management for Multi User Virtual Environments,” PGNET 2009 - Proc. 10th Annu. Postgrad. Symp. Converg. Telecommun. Netw. Broadcast., 2009.
[18] S. Kumar, J. Chhugani, C. K. C. Kim, D. K. D. Kim, a. Nguyen, P. Dubey, C. Bienia, and Y. K. Y. Kim, “Second Life and the New Generation of Virtual Worlds,” Computer (Long. Beach. Calif)., 2008.
[19] K. Getchell, I. Oliver, A. Miller, and C. Allison, “Metaverses as a platform for game based learning,” Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA, pp. 1195–1202, 2010.
[20] “All Messages in Second Life.” [Online]. Available: http://wiki.secondlife.com/wiki/Category:Messages. [Accessed: 01-Apr-2015].
[21] “Second Life Protocol.” [Online]. Available: http://wiki.secondlife.com/wiki/Protocol. [Accessed: 23-Mar-2015].
[22] “Second Life Server Architecture.” [Online]. Available: http://wiki.secondlife.com/wiki/Server_architecture. [Accessed: 28-Mar-2015].
[23] M. Varvello, F. Picconi, C. Diot, and E. Biersack, “Is there life in Second Life?,” Proc. 2008 ACM Conex. Conf. - Conex. ’08, no. April, pp. 1–12, 2008.
[24] C.-A. La and P. Michiardi, “Characterizing User Mobility in Second Life,” Proc. first Work. Online Soc. networks, pp. 79–84, 2008.
[25] H. L. H. Liang, M. Motani, and W. T. O. W. T. Ooi, “Textures in Second Life: Measurement and Analysis,” 2008 14th IEEE Int. Conf. Parallel Distrib. Syst., 2008.
[26] I. Oliver, C. Allison, and A. Miller, “Mongoose: A TCP Fair Second Life client,” Proc. 11th Annu. Postgrad. Symp. Converg. Telecommun. Netw. Broadcast. (PGNet 2010), 2010.
[27] I. Oliver, A. Miller, and C. Allison, “Mongoose: Throughput redistributing virtual world,” 2012 21st Int. Conf. Comput. Commun. Networks, ICCCN 2012 - Proc., 2012.

[28] "Packet Accounting, Second Life Wiki." [Online]. Available: http://wiki.secondlife.com/wiki/Packet_Accounting. [Accessed: 13-Mar-2015].
[29] N. Katz, T. Cook, and R. Smart, "Extending Web Browsers with a Unity 3D-Based Virtual Worlds Viewer," IEEE Internet Comput., vol. 15, no. 5, pp. 15–21, 2011.
[30] "Pixie Viewer for OpenSim." [Online]. Available: http://pixieviewer.com/. [Accessed: 01-Apr-2015].
[31] "Virtual World Framework Main Website." [Online]. Available: https://virtual.wf/documentation.html. [Accessed: 01-Apr-2015].
[32] "Virtual World Framework Demos." [Online]. Available: https://virtual.wf/demos.html. [Accessed: 01-Apr-2015].
[33] "The Virtual World Framework Sandbox." [Online]. Available: https://sandbox.adlnet.gov/904/adl/sandbox/. [Accessed: 02-Apr-2015].
[34] "ReactionGrid Main Website." [Online]. Available: http://reactiongrid.com/. [Accessed: 03-Apr-2015].
[35] M. Belshe and M. Thomson, "Hypertext Transfer Protocol version 2.0 draft-ietf-httpbis-http2-17," 2015.
[36] B. I. Grigorik, "Making the Web Faster," Commun. ACM, vol. 56, no. 12, pp. 1–8, 2013.
[37] D. Stenberg, "HTTP2 explained," ACM SIGCOMM Comput. Commun. Rev., vol. 44, no. 3, pp. 120–128, Jul. 2014.
[38] T. Bryce, R. Jurdak, and I. Atkinson, "SPDYing Up the Web," Commun. ACM, 2012.
[39] P.-H. Kamp, "HTTP/2.0: the IETF is phoning it in," Commun. ACM, vol. 58, no. 3, pp. 40–42, 2015.
[40] I. Grigorik, High Performance Browser Networking - What every web developer should know about networking and web performance. " O'Reilly Media, Inc.," 2013.
[41] J. Khalid, S. Agarwal, A. Akella, and J. Padhye, "Improving the performance of SPDY for mobile devices," hotmobile15, 2015.
[42] R. Peon and H. Ruellan, "HPACK - Header Compression for HTTP/2 draft-ietf-httpbis-header-compression-12," 2015.
[43] Y. Elkhatib, G. Tyson, and M. Welzl, "The Effect of Network and Infrastructural Variables on SPDY's Performance," arXiv Prepr. arXiv1401.6508, pp. 1–23, Jan. 2014.
[44] J. Padhye and H. F. Nielsen, "A comparison of SPDY and HTTP performance," Microsoft Res., 2012.
[45] J. Erman, V. Gopalakrishnan, R. Jana, and K. K. Ramakrishnan, "Towards a SPDY'ier mobile web?," Proc. ninth ACM Conf. Emerg. Netw. Exp. Technol. - Conex. '13, pp. 303–314, 2013.
[46] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, D. Wetherall, and I. Nsdi, "How Speedy is SPDY," Nsdi'14, 2014.
[47] H. Kim, G. Yi, H. Lim, J. Lee, B. Bae, and S. Lee, "Performance Analysis of SPDY Protocol in Wired and Mobile Networks," in Ubiquitous Information Technologies and Applications, Springer, 2014, pp. 199–206.
[48] A. A. Salam, M. Luglio, C. Roseti, and F. Zampognaro, "SPDY multiplexing approach on long-latency links," in Wireless Communications and Networking Conference (WCNC), 2014 IEEE, 2014, pp. 3450–3455.
[49] A. Cardaci and L. Caviglione, "Performance Evaluation of SPDY over High Latency Satellite Channels," Pers. Satell. Serv., pp. 123–134, 2013.
[50] "QUIC Protocol Official Website." [Online]. Available: https://www.chromium.org/quic. [Accessed: 03-Apr-2015].
[51] "QUIC Overview." [Online]. Available: https://docs.google.com/document/d/1gY9-YNDNAB1eip-RTPbqphgySwSNSDHLq9D5Bty4FSU/edit?pli=1. [Accessed: 03-Apr-2015].
[52] "QUIC Wire Layout Specification." [Online]. Available: https://docs.google.com/document/d/1WJvyZflAO2pq77yOLbp9NsGjC1CHetAXV8I0fQe-B_U/edit?pli=1. [Accessed: 03-Apr-2015].
[53] J. Roskind, "QUIC - Multiplexed stream transport over udp - Design Document." [Online]. Available: https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34/mobilebasic?pli=1. [Accessed: 04-Apr-2015].
[54] "Wireshark Main Web page." [Online]. Available: https://www.wireshark.org/.
[55] G. Carlucci, L. De Cicco, and S. Mascolo, "HTTP over UDP: an Experimental Investigation of QUIC," 30th ACM/SIGAPP Symp. Appl. Comput., 2015.
[56] M. Fischlin and F. Günther, "Multi-Stage Key Exchange and the Case of Google's QUIC Protocol," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014, pp. 1193–1204.
[57] R. Lychev, S. Jero, A. Boldyreva, and C. Nita-Rotaru, "How secure and quick is QUIC? Provable security and performance analyses," IEEE Symp. Secur. Priv., 2015.
[58] "QUIC Crypto." [Online]. Available: https://docs.google.com/document/d/1g5nIXAIkN_Y-7XJW5K45IblHd_L2f5LTaDUDwvZ5L6g/edit. [Accessed: 28-Mar-2015].