

Mobility Multihoming Duality for the Internet Protocol

Ryo Yanagida



University of
St Andrews

This thesis is submitted in partial fulfillment for the degree of
Doctor of Philosophy (PhD)
at the University of St Andrews

October 2021

Candidates declaration

I, Ryo Yanagida, do hereby certify that this thesis, submitted for the degree of PhD, which is approximately 41,000 words in length, has been written by me, and that it is the record of work carried out by me, or principally by myself in collaboration with others as acknowledged, and that it has not been submitted in any previous application for any degree. I confirm that any appendices included in my thesis contain only material permitted by the 'Assessment of Post-graduate Research Students' policy.

I was admitted as a research student at the University of St Andrews in February 2017.

I, Ryo Yanagida, received assistance in the writing of this thesis in respect of grammar and spelling, which was provided by Asta Lam.

I received funding from an organisation or institution and have acknowledged the funder(s) in the full text of my thesis.

Date 24/06/2022

Signature of candidate 柳田 涼

Supervisor's declaration

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of PhD in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree. I confirm that any appendices included in the thesis contain only material permitted by the 'Assessment of Postgraduate Research Students' policy.

Date 27 June 2022

Signature of supervisor GJB

Permission for publication

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. We also understand, unless exempt by an award of an embargo as requested below, 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 this thesis will be electronically accessible for personal or research use and that the library has the right to migrate this thesis into new electronic forms as required to ensure continued access to the thesis.

I, Ryo Yanagida, confirm that my thesis does not contain any third-party material that requires copyright clearance.

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

Printed copy

Embargo on all of printed copy for a period of 2 years on the following ground(s):

- Publication would preclude future publication

Supporting statement for printed embargo request

We are currently working on papers including results from this thesis.

Electronic copy

Embargo on all of electronic copy for a period of 2 years on the following ground(s):

- Publication would preclude future publication

Supporting statement for electronic embargo request

We are currently working on papers including results from this thesis.

Title and Abstract

- I agree to the title and abstract being published.

Date 24/06/2022

Signature of candidate

Date 27 June 2022

Signature of supervisor

Underpinning Research Data or Digital Outputs

Candidate's declaration

I, Ryo Yanagida, hereby certify that no requirements to deposit original research data or digital outputs apply to this thesis and that, where appropriate, secondary data used have been referenced in the full text of my thesis.

Date

24/06/2022

Signature of candidate

柳田 涼

Abstract

In the current Internet, mobile devices with multiple connectivity are becoming increasingly common; however, the Internet protocol itself has not evolved accordingly. Instead, add-on mechanisms have emerged, but they do not integrate well. Currently, the user suffers from disruption to communication on the end-host as the physical network connectivity changes. This is because the IP address changes when the point of attachment changes, breaking the transport layer end-to-end state. Furthermore, while a device can be connected to multiple networks simultaneously, the use of IP addresses prevents end-hosts from leveraging multiple network interfaces — a feature known as host multihoming, which can potentially improve the throughput or reliability. While solutions exist separately for mobility and multihoming, it is not possible to use them as a duality solution for the end-host.

This work extended ILNPv6, an engineering solution of Identifier-Locator Network Protocol (ILNP) implemented as a superset of IPv6 on the Linux kernel. The existing implementation was extended to enable mobility and multihoming duality. First, the mobility implementation was enhanced to support continuous mobility; a comparative analysis against Mobile IPv6 (MIPv6) showed superior performance during a series of handoffs. Second, multihoming was implemented and integrated with mobility; the evaluation with a flexible multi-connectivity scenario with load-balancing showed negligible loss and consistent throughput. Finally, the impact of the combined mobility-multihoming mechanism was evaluated with a real-time video streaming application showing continuous uninterrupted real-time video playback up to 2160p (4K ultra high definition). Overall, this work has demonstrated that mobility-multihoming duality is possible for end-hosts over IPv6 for existing applications without changing the network infrastructure.

Acknowledgements

Many people have greatly contributed to completing this thesis.

First and foremost, I would like to extend my deepest gratitude to **Professor Saleem Bhatti**. Without his guidance, support, wisdom, and more, it would not have been possible to complete this thesis. His support has been vital in many aspects of my PhD. I am incredibly grateful to him for giving me this opportunity to learn so much.

Second of all, I am incredibly grateful to have my family's support from the beginning. I cannot thank them enough for their love, care, and encouragement.

I am extremely grateful to have **Dr Tristan Henderson** as my second supervisor. I am thankful for all the suggestions and pointers that he gave me at various points of my PhD. Discussions with him have kept me grounded, reminding me of what is important.

Special thanks to **Dr Colin Allison**, for various pointers and proofreading my thesis. I would also like to extend my gratitude towards many staff members of the School of Computer Science, who provided various advice and pointers all stages of my PhD.

I am deeply indebted to my best friend and partner **Asta Lam**. Her encouragement and support have been vital to keep me going in difficult times. Your eagle eyes have saved me many times during the proofreading stage of the thesis.

I would like to extend my sincere thanks to my friends who believed in me and kept me going, including **Xu Zhu, Janis Wong, Donald Robertson, Alice Lynch**, and many others. I have been supported by so many of you that it would be impossible for me to write all of your names down on this page.

Many thanks to **Khawar Shehzad, Ditchaphong Phoomikiattisak, Simone Conte**, and **Tom Dalton** for various discussions and pointers, and for sharing various resources.

Special thanks to the Systems Team at the School of Computer Science. I am incredibly grateful to them for their help throughout my research with equipment, valuable suggestions, and more.

Huge thank you to the Admin Team at the School of Computer Science for helping me navigate various situations, making it possible for me to focus on my research.

Funding

I would like to thank **Time Warner Cable (TWC)**, now owned by **Charter Communication**, for partially funding my PhD.

Contents

Abstract	i
Acknowledgements	ii
Contents	iv
Acronyms	xii
List of Figures	xviii
List of Tables	xxxiii
List of Publications	xxxvi
1 Introduction	1
1.1 Host mobility	2
1.2 Host multihoming	2
1.3 Host mobility-multihoming duality	3
1.4 Importance of the network layer, end-host-only solution	3
1.5 Thesis statement	5
1.6 Thesis structure	6
2 Background	8
2.1 History of IP networking and previous attempts to enable identifier-locator split	9
2.1.1 Internet Protocol and its naming architecture	9
2.1.2 Known issues in the naming architecture in the Internet — The entanglement of the name spaces	11
2.2 Historic Identifier-Locator split architectures	12
2.2.1 8+8/GSE draft by Mike O'Dell	12

2.2.2	Nimrod architecture	13
2.3	Mobility options across different technologies	14
2.3.1	Celluar technologies	15
2.3.2	IEEE 802.11	15
2.4	Multihoming options at the lower layer	16
2.5	Current and previous solutions that approach mobility and multihoming separately	16
2.6	Mobility solutions	17
2.6.1	Shim layer approach	18
2.6.2	Network solutions	18
2.6.3	Transport layer solution	21
2.7	Multihoming solutions	21
2.7.1	Shim layer approach	21
2.7.2	Network solutions	22
2.7.3	Transport layer solutions	23
2.8	Current approaches to mobility-multihoming	24
2.8.1	Comparison criteria	25
2.9	Host mobility and multihoming solutions	27
2.9.1	Network solution	27
2.9.2	End-host based shim layer approach	27
2.9.3	End-host based re-naming approach	28
2.9.4	Transport layer approach	28
2.10	Current available solutions	29
2.11	Qualitative analysis of MP-TCP	30
2.11.1	Results	31
2.11.2	Discussion	31
2.11.3	Summary	34
2.12	End-to-end principle and ILNP	34
2.13	Networked applications on the Internet	35
2.14	Summary	36

3	Introduction to Identifier Locator Network Protocol	37
3.1	Identifier Locator Network Protocol (ILNP) — Architecture	37
3.1.1	Locator Update (LU)	38
3.2	Identifier Locator Network Protocol v6 (ILNPv6) — Engineering solution	40
3.2.1	NID and L64 in IPv6 ‘wire-image’	40
3.3	Previous work on Identifier Locator Network Protocol v6 (ILNPv6)	41
3.3.1	IP host mobility with ILNPv6 on Linux kernel 3.9	42
3.3.2	IP site multihoming with ILNPv6 on FreeBSD	42
3.4	ILNPv6 on Linux kernel 4.9.52	43
3.4.1	Locator Update (LU) with ILNPv6	44
3.4.2	Interface configuration for ILNPv6 hosts	44
3.4.3	ILNP Communication Cache (ILCC)	44
3.4.4	ILNPv6 Nonce	45
3.5	Consolidated summary and comparison against existing mobility and multihoming mechanisms	45
3.6	Summary	48
4	Testbed and kernel development to enable empirical evaluation	50
4.1	Problem space	50
4.2	Rationale for testbed-based evaluation and engineering effort	53
4.3	Development testbed	53
4.4	Evaluation testbed	54
4.4.1	Hardware and software	55
4.4.2	Network	59
4.5	Evaluation scenarios	60
4.5.1	Continuous mobility scenario	61
4.5.2	Mobility-multihoming duality scenario	61
4.5.3	Executing evaluation scenarios	63
4.5.4	Emulating movement	63
4.5.5	Overlap duration	64

4.5.6	Path round trip time (RTT) configuration	66
4.5.7	Payloads	67
4.6	Data collection and metrics	68
4.7	Software engineering summary	70
4.7.1	Existing files modified in the Linux kernel	71
4.8	Summary	73
5	Continuous Mobility	74
5.1	Changes from previous work on Identifier Locator Network Protocol v6 (ILNPv6)	74
5.1.1	Scientific contributions	74
5.1.2	Changes from previous ILNPv6 implementation in Linux kernel	75
5.1.3	Multiple handoffs	76
5.1.4	Locator Update and L64 state	76
5.1.5	Socket lookup	78
5.1.6	Interface selection for NID generation	78
5.2	Evaluation Setup	80
5.3	Procedure	80
5.3.1	Emulating Movement	81
5.3.2	Metrics	82
5.3.3	Data collection and analysis	83
5.4	Results	84
5.4.1	TCP iPerf2	85
5.4.2	UDP iPerf2	87
5.4.3	Results summary	89
5.5	Discussion	89
5.5.1	IPv6 Router Advertisement in continuous handoff scenario	89
5.5.2	Wireless versus wired technology	93
5.5.3	Handoff latency	94
5.5.4	Buffering effect in TCP scenario	96
5.5.5	Round trip time (RTT) emulation	97

5.5.6	Impact of interface selection mechanisms using route metric	98
5.5.7	Omitted L64 state — AGED	98
5.6	Summary	99
6	Mobility and Multihoming Duality	100
6.1	Overview of mobility-multihoming duality mechanism in ILNPv6	102
6.2	Changes to ILNPv6 and Linux kernel to enable mobility and multihoming duality	103
6.2.1	Route selection mechanism and transport layer packet processing	105
6.2.2	Load-sharing	108
6.2.3	Extension to the Locator Update message in ILNPv6 implementation . . .	108
6.2.4	New Locator Update message in draft RFC	114
6.2.5	Soft-handoff in multihoming	116
6.2.6	Sysctl interfaces used to control ILNPv6 mobility-multihoming duality mechanism	117
6.3	Evaluation of continuous multihomed-mobility scenario	118
6.3.1	Continuous multihomed-mobility scenario	118
6.3.2	Experiment setup	119
6.3.3	Procedure	120
6.4	Results	121
6.4.1	Example runs	122
6.4.2	Aggregated results	122
6.5	Discussion	126
6.5.1	Interface management in a mobile-multihomed host	126
6.5.2	Fixed behaviour in mobility-multihoming duality load-sharing application implementation	128
6.5.3	Boot-strapping multihoming transport	130
6.6	Summary	131
7	Application level impact on real-time video	132
7.1	Rationale for QoE analysis of real-time video application over ILNPv6	132
7.2	ILNPv6 backwards compatibility and legacy application support	133

7.3	Impacts of network layer performance on application layer performance	134
7.4	‘Real-time’ video transport experiment setup	134
7.4.1	Video payload selection and preparation	135
7.4.2	Added delays to round trip time (RTT)	137
7.4.3	Metrics	138
7.4.4	Data collection and analysis	141
7.4.5	Testbed	142
7.4.6	Reference point	143
7.5	‘Real-time’ video transport evaluation with continuous mobility	144
7.5.1	Scenario	144
7.5.2	Results	145
7.6	‘Real-time’ video transport experiment with multihoming mobility duality	153
7.6.1	Scenario	153
7.6.2	Results	153
7.7	Discussion	161
7.7.1	‘Real-time’ video transfer as a <i>real-life, real-time</i> application and its design space	161
7.7.2	‘Real-time’ video scenario with pre-recorded clips	162
7.8	Summary	162
8	Conclusion	164
8.1	Contributions	164
8.2	Discussion	165
8.2.1	Lack of existing network layer host mobility-multihoming duality solutions	165
8.2.2	Effects on TCP congestion control	166
8.3	Future work	166
8.3.1	Path selection and management mechanism	166
8.3.2	Potential privacy enhancements with NID values	167
8.3.3	New transport protocol and ILNPv6	167
8.3.4	Enabler of ubiquitous computing	168

Appendices	169
A ILNPv6 implementation limitation	169
A.1 Multiple interfaces present on the same IPv6 subnet	169
A.2 Multiple Node Identifier (NID)s	170
B Experimental values	171
B.1 Introduction	171
B.2 Overlap duration	171
B.3 Round trip time (RTT) selection	171
B.3.1 ‘National’ RTT traces	172
B.3.2 ‘Intercontinental’ RTT traces	174
C Multihoming implementation decisions	178
D Behaviours observed in the evaluation testbed	180
D.1 Interaction between TCP window scaling, iperf, and throughput visualisation in high-latency links	180
E Real time protocol (RTP) experiment and video metrics	182
E.1 VQMT analysis	182
E.2 Acquiring and preparation of the video samples	184
F Enlarged plots	185
F.1 Continuous mobility typical runs	185
F.2 Mobility-multihoming duality typical runs	190
F.3 Real-time video continuous mobility typical runs	199
F.4 Real-time video mobility-multihoming duality runs	204
F.5 MultiPath-TCP (MP-TCP) example runs	209
G Hardware used in evaluation	212
G.1 Network equipment	212
G.1.1 Ubiquiti Networks EdgeRouter X	213

G.1.2 Ubiquiti Networks EdgeRouter 6P	214
G.1.3 Ubiquiti Networks Edge OS	215
G.2 End-hosts	216

Bibliography	221
---------------------	------------

Acronyms

API application programming interface.

BBB Big Buck Bunny.

BBC British Broadcasting Company.

BEX HIP Base Exchange.

BU Binding Update.

BU-Ack Binding Update Acknowledgement.

CN Correspondent Node.

CoA Care-of Address.

CoT Care-of Test.

CoTI Care-of Test Init.

CSV comma-separated values.

CWND congestion window.

DAD duplicate address detection.

DHCP Dynamic Host Configuration Protocol.

DIME Dynamic Internet Mobility for End-Systems.

DMM Distributed Mobility Management.

DNS domain name system.

DoS denial of service.

DRR deficit round-robin.

EID End-System Identifier.

ESD End System Designator.

ETR Egress Tunnel Router.

FA Foreign Agent.

FMIPv6 Fast-handover for Mobile IPv6.

FQDN fully qualified domain name.

HA Home Agent.

HD high definition.

HI Host Identifier.

HID Host Identifier.

HIP Host Identity Protocol.

HIT Host Identifier Tag.

HMIPv6 Hierarchical Mobile IPv6.

HoA Home Address.

HoT Home Test.

HoTI Home Test Init.

HTTP Hyper Text Transfer Protocol.

HTTPS Hyper Text Transfer Protocol Secure.

I-LV Identifier-Locator Vector.

ICMPv6 Internet Control Message Protocol v6.

IEEE Institute of Electrical and Electronics Engineers.

IETF Internet Engineering Task Force.

IHMP Internet Host Mobility Protocol.

IID interface identifier.

ILCC ILNP Communication Cache.

ILNP Identifier Locator Network Protocol.

ILNPv6 Identifier Locator Network Protocol v6.

IP Internet Protocol.

IPAT inter-packet arrival time.

IRTF Internet Research Task Force.

ISP Internet service provider.

ITR Ingress Tunnel Router.

L Locator.

L64 Locator.

LAN local area network.

LISP Locator Identifier Separation Protocol.

LISP-MN Locator Identifier Separation Protocol Mobile Node.

LKM loadable kernel module.

LMA Local Mobility Anchor.

LTS long-term support.

LU Locator Update.

LU-Ack Locator Update Acknowledgement.

MAC media access control.

MAG Mobility Access Gateway.

MANEMO Mobile Ad-hoc Network Mobility.

MANET mobile ad-hoc network.

MHMP Multihomed with MultiPrefix.

MIP Mobile IP.

MIPv4 Mobile IPv4.

MIPv6 Mobile IPv6.

MIS Media Independent Service.

MITM man-in-the-middle.

MN Mobile Node.

MP-TCP MultiPath-TCP.

MPEG-TS MPEG Transport Stream.

MVNO Mobile Virtual Network Operator.

NAT network address translation.

NEMO Network Mobility.

NID Node Identifier.

NPTv6 IPv6 Network Prefix Translation.

OS operating system.

PETR Proxy Egress Tunnel Router.

PITR Proxy Ingress Tunnel Router.

PMIPv6 Proxy Mobile IPv6.

PSNR peak signal-to-noise ratio.

QoE Quality of Experience.

QoS Quality of Service.

RA router advertisement.

RFC Request For Comments.

RG Routing Goop.

RLOC Routing Locator.

RO Route Optimisation.

RS router solicitation.

RSSI received signal strength indicator.

RTO retransmission timeout.

RTP Real-time Transport Protocol.

RTT round trip time.

SCTP Stream Control Transmission Protocol.

SHIM6 Level 3 Multihoming Shim Protocol for IPv6.

SLAAC stateless address autoconfiguration.

SNR signal-to-noise-ratio.

SoC system on a chip.

SPoF single point of failure.

SSID service set identifier.

SSIM structural similarity index.

TCP Transmission Control Protocol.

ToS Tears of Steel.

UDP User Datagram Protocol.

ULID Upper-layer Identifier.

URL uniform resource locator.

UX user-experience.

VBR variable bit rate.

VLAN virtual LAN.

VoIP Voice over IP.

VQMT Video Quality Measurement Tool.

WAN wide area network.

WG working group.

Wi-Fi Wireless-Fidelity.

WLAN wireless LAN.

List of Figures

1.1	Diagrams illustrating the issues of providing additional network functionalities at the layers above and under the network layer (layer 3). (a) illustrates the complexity in enabling inter-workings individually at layer 2. The arrows illustrate the distinct solution for enabling mobility from and to the respective connectivities. (b) illustrates the narrow scopes of mechanisms built into the transport layer. Any mechanism at the transport layer is only applicable to applications using the specific transport layer protocol. Even with the ‘same’ transport protocol — i.e. TCP vs MP-TCP — any additional feature often requires a modified application to make use of it.	4
1.2	A diagram illustrating the wide scope of the impact from the solution at the network layer. A solution implemented at the network layer enables the inter-workings of different technologies at the lower layers, while still providing functionalities to different transport layer protocols and all applications making use of these transport layer protocols.	5
2.1	Throughput facet plot of MP-TCP flow received at the Mobile Node (MN). The top three plots show the throughputs received at the addresses of the respective three interfaces at the MN and the bottom plot shows the aggregate throughput. The distribution of the throughput is uneven and changes to the throughput on the individual interfaces are ‘bursty’. The vertical line shows the protocol level signalling (here, MP-TCP specific multihoming management signal) to add or remove a connectivity received at the respective addresses.	32

2.2	Throughput facet plot of MP-TCP flow received at the Correspondent Node (CN). The top three plots show the throughputs received from the addresses of the respective three interfaces at the MN and the bottom plot shows the aggregate throughput. The distribution of the throughput is uneven and changes to the throughput on the individual interfaces are ‘bursty’. The vertical line shows the protocol level signalling (here, MP-TCP specific multihoming management signal) to add or remove a connectivity received from the respective addresses.	33
3.1	Timeline diagram of a Locator Update (LU) messages: In order for an MN to update its CN with its changes to network connectivity, an Locator Update (LU) is sent; the CN then updates its ILNP Communication Cache (ILCC) accordingly, and then replies with a Locator Update Acknowledgement (LU-Ack). This only requires one round trip time (RTT).	39
3.2	A diagram showing address/IL-V formats and the sizes of each part for the respective network protocols [1]. ILNPv6 enables identifier-locator split using the already existing prefix-IID split of the IPv6 unicast addressing.	41
3.3	A diagram comparing packet headers for IPv6 and ILNPv6 packets [1]. The ‘wire-image’ of ILNPv6 packets, in terms of address, as shown, lines up with the IPv6 global unicast prefix and IID. For ILNPv6, a nonce follows, which is added using the Destination Option Header, as defined in Section 4.6 of RFC8200 [2].	42
3.4	The L64 state diagram describing how the L64 state transitions. The lists below the state diagram define the states and actions.	46
4.1	A diagram illustrating a potential scenario for the use case of a mobile device. A user begins a communication session at home via 802.11 Wireless-Fidelity (Wi-Fi), passes through different network connectivities, and arrives at their place of work, which also provides 802.11 Wi-Fi.	51

4.2	A photo of the testbed. From the left side: m5 (CN), R4, R1–R3, and m4 (MN). The switch at the back was purely for control and management of the testbed. Orange and black cables were used for the management network only. The yellow ethernet cables were for network aa , the blue ethernet cables were for network bb , the red ethernet cables were for network cc , and the green ethernet cable was for the CN, network dd . The HA for MIPv6 was outside of the frame on the shelf just behind the CN. The physical hardware of HA was the same as the CN and the MN.	57
4.3	A physical topology diagram of the evaluation testbed.	59
4.4	A scenario diagram for continuous mobility evaluation. The movement of the MN is shown with the green arrow. The host will move across networks served by R1–R3, and back to R1 again continuously until the communication ends. . . .	62
4.5	A scenario diagram describing host movement for the mobility-multihoming duality scenario. The arrow labelled 1 is the first movement the MN carries out: moving from network aa on R1 to network cc on R3. The arrow labelled 2 shows the second set of movement where the MN returns from network cc back to network aa.	62
4.6	A diagram illustrating an overlap duration and events during the overlap duration.	65
5.1	An L64 state diagram showing how the L64 state transitions without the AGED state.	79

5.2	The above plots show the throughput and the sequence numbers observed over time in the iperf tcp flow scenario. The throughput is shown in each facet plot: the top three plots in (a) and (b) show throughput received at the MN, at the given prefix/Locator (L64), and the bottom plot shows the aggregate throughput. Plots (c) and (d) show the sequence numbers observed over the duration of the flow. The vertical dashed lines indicate mobility signalling packets: LU for ILNPv6; and Care-of Test Init (CoTI), Care-of Test (CoT), Binding Update (BU), and Binding Update Acknowledgement (BU-Ack) for Mobile IPv6 (MIPv6). As shown in (a), MIPv6 experienced ‘gaps’ in data transmission, followed by bursts in throughput. This is also evident in the sequence numbers shown in (c), where the sequence numbers dramatically increased vertically, i.e. almost instantaneously, indicating a burst in transmission. Meanwhile, ILNPv6 experienced a stable, consistent flow of data with consistent increase in sequence numbers.	86
5.3	The plots show the ‘gaps’ in throughput. (a) shows the cumulative duration of ‘gaps’ in throughput in each 120 seconds flow. (b) shows the maximum and the median of the individual ‘gaps’ observed in 120 seconds flow. The × indicates the mean.	87
5.4	The plots show the distribution of observed throughput in 0.5 second intervals for TCP iPerf2 flow. While ILNPv6 experienced most of the throughput at 10 Mbps, MIPv6 experienced a wide range of throughput down to 0 Mbps. Note that MIPv6 TCP results (labelled mip6-tcp) has outliers up to 150 Mbps due to the TCP buffering effects from the testbed, which were omitted for the clarity of the result. The table underneath describes the interquartile range.	88

5.5	Throughput and sequence numbers plotted over time for MIPv6 and ILNPv6 iperf User Datagram Protocol (UDP) continuous mobility scenarios. The vertical dashed lines indicate mobility signalling packets: LU for ILNPv6; and CoTI, CoT, BU, and BU-Ack for MIPv6. Similar to the Transmission Control Protocol (TCP) mobility scenario, MIPv6 experienced significant gaps of throughput during handoffs. As UDP does not buffer or resend packets, loss of throughput during handoff directly led to loss of packets evident in the gaps in sequence numbers observed. Meanwhile, ILNPv6 results showed a much more stable aggregate throughput, and no gaps in the sequence numbers delivered.	90
5.6	The plot showing the distribution of observed throughput for UDP iperf flow. While ILNPv6 achieved stable throughput around 10 Mbps, MIPv6 had a wider distribution extending all the way down to 0 Mbps. As the aggregates were sampled across the whole run, it was evident that ILNPv6 experienced throughput between 0 Mbps to 10 Mbps at the beginning or at the end when some packets spilled over the 0.5 second aggregation buckets. The table below describes the interquartile ranges.	91
5.7	A box plot showing the packet loss observed in the iperf UDP flows. Compared to the ILNPv6 flows, MIPv6 suffered a much higher number of packets dropped, approximately 4–6 times more. The × indicates the mean.	91
5.8	A handoff message timeline for MIPv6 with route optimisation. The handoff is triggered by a loss of communication. The handoff requires well beyond 1 RTT, due to the number of messages required.	95

6.1	A timeline diagram showing an example of mobility-multihoming duality scenario. The MN has three interfaces and the CN has one interface. The MN and the CN begin a communication session using the single interface on both sides. The MN activates interface 2, receiving prefix <code>bb</code> , sending an LU and setting the new locator as <code>ACTIVE</code> . The CN responds with an LU-Ack accordingly, acknowledging the new locator set. The MN continues to activate another interface, which is handled accordingly by the CN as well. The MN then lists the first interface in the <code>ilnp_disabled_interface</code> sysctl list, triggering another LU and state changes in the ILCC. After the CN acknowledges the removal of the first interface, the MN removes the interface.	104
6.2	A flowchart describing UDP/TCP packet processing with ILNPv6 mobility-multihoming duality mechanism with Deficit Round-Robin (DRR) load-sharing. Boxes coloured grey indicate unmodified processes. Yellow boxes indicate modifications. Green boxes are the additional mechanism in <code>ilnp6.c</code>	107
6.3	A callgraph of the UDP ingress processing path. Grey boxes indicate unmodified functions, yellow boxes indicate modified existing functions, blue-grey boxes indicate references to the functions such as pointers, cyan boxes indicate ILNPv6-specific functions in the ILNPv6 file , and boxes with a dotted outline indicate new functions for implementing the mobility-multihoming duality mechanism.	109
6.4	A callgraph of the UDP egress processing path. Grey boxes indicate unmodified functions, yellow boxes indicate modified existing functions, blue-grey boxes indicate references to the functions such as pointers, cyan boxes indicate ILNPv6-specific functions in the ILNPv6 file , and boxes with a dotted outline indicate new functions for implementing the mobility-multihoming duality mechanism.	110

6.5	A callgraph of the TCP ingress processing path. Grey boxes indicate unmodified functions, yellow boxes indicate modified existing functions, blue-grey boxes indicate references to the functions such as pointers, cyan boxes indicate ILNPv6-specific functions in the ILNPv6 file , and boxes with a dotted outline indicate new functions for implementing the mobility-multihoming duality mechanism.	111
6.6	A callgraph of the TCP egress processing path. Grey boxes indicate unmodified functions, yellow boxes indicate modified existing functions, blue-grey boxes indicate references to the functions such as pointers, cyan boxes indicate ILNPv6-specific functions in the ILNPv6 file , and boxes with a dotted outline indicate new functions for implementing the mobility-multihoming duality mechanism.	112
6.7	The ILNPv6 LU message structure based on the message format from RFC6743 [3].	113
6.8	The ILNPv6 LU message structure currently being drafted.	115
6.9	A scenario diagram describing the MN's movement for mobility-multihoming duality scenario. The arrow labelled 1 is the first movement that the MN carries out: moving from network aa on R1 to network cc on R3. The arrow labelled 2 shows the second set of movement where the MN returns from network cc back to network aa	120
6.10	Plots showing the throughput and sequence numbers observed in typical mobility-multihoming duality iperf2 TCP scenario evaluations. Plots (a) and (c) represent the flow received at the CN. Plots (b) and (d) represent the flow received at the MN. The results showed consistent aggregate throughput at the bottom facet of plots (a) and (b) , while the top three facets show throughput observed at the respective source/destination IL-V, showing smooth, equal splits. The vertical dashed line shows the Locator Update messages. Plots (c) and (d) show consistent increase in the sequence numbers, indicating a consistent flow and delivery of packets.	123

6.11	Plots showing the throughput and sequence numbers observed in typical mobility-multihoming duality iperf2 UDP scenario evaluations. Plots (a) and (c) represent the flow received at the CN. Plots (b) and (d) represent the flow received at the MN. The results showed consistent aggregate throughput at the bottom facet of plots (a) and (b) , while the top three facets show throughput observed at the respective source/destination IL-V, showing smooth, equal splits. The vertical dashed line shows the Locator Update messages. Plots (c) and (d) show consistent increase in the sequence numbers, indicating a consistent flow and delivery of packets.	124
6.12	A plot showing the TCP misorder ratio for the sequence numbers and the acknowledgement numbers received. Little to no misordering was observed with both the sequence numbers and the acknowledgement numbers received. The × indicates the mean.	125
6.13	A plot showing the TCP resend ratio for the sequence numbers and acknowledgement received. No significant resend was observed for both the sequence numbers and the acknowledgement numbers. The × indicates the mean.	125
6.14	A plot showing the TCP throughput observed in the mobility-multihoming duality scenarios with the iperf TCP flow over ILNPv6. Across all delay scenarios, the throughput remained near the 10 Mbps target with negligible distribution. The × indicates the mean.	126
6.15	A plot showing the UDP packet statistics observed in mobility-multihoming duality scenarios with the iperf UDP flow over ILNPv6. With all scenarios, both misordering and loss rate remained little to none. The × indicates the mean. . .	127
6.16	A plot showing the throughput observed in mobility-multihoming duality scenarios with the iperf UDP flow over ILNPv6. The throughput remained consistent at around the 10 Mbps target across the board with very few exceptions. The × indicates the mean.	127
7.1	A screenshot from Microsoft Teams during a call in the ‘call health’ pane, indicating that H.264 software encoding is used for the video.	137

7.2	A figure describing relationships between MSE and SSIM with regards to different distortions applied to an image in [4]. Example D in particular has a negative structural similarity index (SSIM); in this image, the local image structure is inverted.	140
7.3	The physical topology diagram of the ILNPv6 testbed.	143
7.4	Plots showing reference throughput with a fixed connection. (a) shows the throughput plot of the 1080p Tears of Steel clip without any added RTT.(b) shows the throughput plot of the 1080p Big Buck Bunny clip without any added RTT.The width of the plots is matched approximately with the following example run plots.	144

7.5 The two plots on the left are from a ‘typical run’ with MIPv6. The two plots on the right are from a ‘typical run’ with ILNPv6. They are both runs from the 1080p Tears of Steel scenarios. The top two graphs represent the SSIM observed in the individual frames from the 0th to 2879th frames (120s at 24 fps, totalling 2880 frames). The bottom two graphs show the throughput observed during the transmission of the videos over respective network protocols to individual network interfaces, as well as the aggregate of the throughput. The throughput on the individual networks fluctuates as the node moves from one to another. The bottom graph labelled aggregate is the total throughput of all three networks at a given time on the x-axis. While the bottom graphs and the top graphs have different units on the x-axis, they are correlated — the top graphs are the resultant quality of the video transmitted at the rate shown in the bottom graphs. SSIM is a common metric for video quality, where 1.0 indicates that the footage is identical to its original footage with no quality degradation. Anything lower than 1.0 indicates loss of quality/mismatched image. The video was encoded as variable bit rate, which resulted in the fluctuations in aggregate throughput. As the same footage was transmitted using the same application, only with different network layer protocols, the aggregate throughput should look the same. However, in the MIPv6 results, the aggregate throughput seemed to have significant intervals of duration with 0 Mbps throughput, where no data was received by any interfaces. While on ILNPv6, there were no such gaps in data transmission. The ILNPv6 SSIM plot shows no changes over time; unlike the ILNPv6 result, the MIPv6 SSIM plot shows large changes of SSIM away from 1.0 for the same number of occurrences as the throughput gaps in the aggregate plot. Since there were no large spikes of throughput to recover the data not received in those durations, there was likely a loss. In fact, the next page shows sequence numbers on RTP packets received across time series supporting this observation. 146

7.6 The two plots on the left are from a typical run with MIPv6. The two plots on the right are from a typical run with ILNP. They are both from the 1080p Tears of Steel scenarios identical to the runs in the previous set of graphs. The top two graphs represent the SSIM observed in the individual frames from the 0th to 2879th frames (120s at 24 fps, totalling 2880 frames). The bottom two graphs are RTP sequence number graphs during the transmission of the video over respective network protocols. As RTP sequence numbers start from a random number, for the purpose of plotting, the first sequence number was subtracted from the sequence numbers. Similar to the previous set of plots, the top graphs show the resultant quality of video received with the sequence numbers shown on the bottom graphs with respective timing. This set of graphs are also from the same set of runs in the previous page. On the bottom-left graph, it is evident that as the time passed, some of the sequence numbers were missing, which suggests those specific packets carrying the video were lost. Meanwhile, the ILNPv6 result on the right does not have any gaps, which suggests that there was no loss. This is in line with the previous result with 0 throughput durations on the MIPv6 run. 147

7.7 A series of frames captured after converting the received MPEG-TS files to raw YUV 4:2:0 files were used for analysis using VQMT. Images on the left column are frames captured from ILNP results. Images on the right column are frames captured from MIPv6 runs. **(a) and (b)** are the 2256th frame of the recorded clip. Frame (b), from the MIPv6 result, shows clear block noise and overall delay in image transition. **(c) and (d)** are the 1089th frame into the clip. Frame (d) from MIPv6 shows clear colour noise and block noise. Note that the part of the robotic arm with a needle similar to the one seen in (c) is visible, which is from the expected frame. The more dominant image in frame (d) is a man with glasses from the previous scene. **(e) and (f)** are the 1325th frame — they were mismatched. Frame (f) is visually not impaired, but it is not the expected frame at that time, which should be what (e) shows. Upon playing back the MPEG2-TS file, at the same timecode, frame (e) was displayed with the ILNP clip, but for the MIPv6 clip, the player displayed the identical frame shown in (f) for multiple seconds. . 149

7.8	Box plots of the mean SSIM and delivery rate values observed for the 120s clip streamed over MIPv6 and ILNP, recorded at the MN. ToS refers to Tears of Steel, a live action sci-fi movie with 3D effects; BBB refers to Big Buck Bunny, a 3D animation movie. An SSIM value of 1.0 indicates being identical to the reference image. The SSIM values are calculated frame by frame against the original clip used for streaming and the mean SSIM is calculated by taking the average throughout a run. The delivery rate values are calculated by observing the RTP sequence numbers from the packet captures.	150
7.9	Box plots of the mean SSIM and delivery rate values observed for the 120s clip streamed over MIPv6 and ILNP, recorded at the MN. The network has an additional 20ms RTT between the MN and the CN to emulate the wider topological/physical distance between them. ToS refers to Tears of Steel, a live action sci-fi movie with 3D effects; BBB refers to Big Buck Bunny, a 3D animation movie. An SSIM value of 1.0 indicates being identical to the reference image. The SSIM values are calculated frame by frame against the original clip used for streaming and the mean SSIM is calculated by taking the average throughout a run. The delivery rate values are calculated by observing the RTP sequence numbers from the packet captures.	151
7.10	Box plots of the mean SSIM and the delivery rate values observed for the 120s clip streamed over MIPv6 and ILNP, recorded at the MN. The network has an additional 200ms RTT between the MN and the CN to emulate the wider topological/physical distance between them. ToS refers to Tears of Steel, a live action sci-fi movie with 3D effects; BBB refers to Big Buck Bunny, a 3D animation movie. An SSIM value of 1.0 indicates being identical to the reference image. The SSIM values are calculated frame by frame against the original clip used for streaming and the mean SSIM is calculated by taking the average throughout a run. The delivery rate values are calculated by observing the RTP sequence numbers from the packet captures.	152

- 7.11 Example plots showing the SSIM per frame and throughput over time for the two clips: Tears of Steel and Big Buck Bunny. **(a) and (c)** show the result from Big Buck Bunny, and **(b) and (d)** show the result from Tears of Steel. They are both 1080p clips with no added delay in the network. The top plot shows the SSIM per frame — SSIM is the structural similarity of the captured clips to the original clip. The flat line at 1.0 indicates that the received image is identical to the original in the context of SSIM. The facet throughput plots show the throughput received at the respective three L64s on three interfaces in the top three plots, with the aggregate of them at the bottom of the facet plots. Dashed lines show the LU/LU-Ack received/sent from the respective interface. The aggregate throughputs here are very similar to the reference static IPv6 single home run shown in Figure 7.7. 155
- 7.12 Example plots showing the SSIM per frame and the sequence numbers observed over time for the two clips: Tears of Steel and Big Buck Bunny. They are both at 1080p resolution. Plots **(a) and (c)** show results from Big Buck Bunny and **(b) and (d)** show results from Tears of Steel. **(a) and (b)** show the SSIM per frame with both cases indicating 1.0 throughout the flow. In the context of SSIM, 1.0 indicates that the original and the received images are identical. The sequence numbers are observed from the RTP packet header. The continuous plot without any gaps indicates that there were no significant loss in data transfer. Also, there are no noticeable amount of misordering visible in the plot, which would have been indicated by diverging ‘lines’. 156
- 7.13 Boxplots showing the SSIM stats for the mobility-multihome duality RTP flows. As shown, both the median and the mean stay close to 1.0, indicating little to no loss of quality for a large proportion of time. In some cases, especially for Big Buck Bunny, for longer RTT, some quality degradation were observed, which are shown as slightly lower mean SSIM. However, provided that both the median and the mean still remain at near 1.0, only a very small portion of the clip experienced quality degradation. The × indicates the mean. 158

7.14	Boxplots showing the RTP packets stats for mobility-multihome duality RTP flows. As the RTT increased, the misordering observed increased. The loss remained little to none. As these were RTP/UDP flows, no duplicates were observed as expected. The × indicates the mean.	159
B.1	A boxplot showing the distribution of the time taken for the layer-3 IPv6 connectivity to become available from the time the interface was enabled. The measurement was taken for 100 runs.	172
F.1	Enlarged plots of continuous mobility MIPv6 TCP typical run throughput. . . .	186
F.2	Enlarged plot of continuous mobility MIPv6 TCP typical run sequence numbers. . . .	187
F.3	Enlarged plots of continuous mobility ILNPv6 TCP typical run throughput. . . .	188
F.4	Enlarged plot of continuous mobility ILNPv6 TCP typical run sequence numbers. . . .	189
F.5	Enlarged plots of mobility-multihoming duality ILNPv6 TCP typical run throughput at the CN.	191
F.6	Enlarged plot of mobility-multihoming duality ILNPv6 TCP typical run sequence numbers at the CN.	192
F.7	Enlarged plots of mobility-multihoming duality ILNPv6 TCP typical run throughput at the MN.	193
F.8	Enlarged plot of mobility-multihoming duality ILNPv6 TCP typical run sequence numbers at the MN.	194
F.9	Enlarged plots of mobility-multihoming duality ILNPv6 UDP typical run throughput at the CN.	195
F.10	Enlarged plot of mobility-multihoming duality ILNPv6 UDP typical run sequence numbers at the CN.	196
F.11	Enlarged plots of mobility-multihoming duality ILNPv6 UDP typical run throughput at the MN.	197
F.12	Enlarged mobility-multihoming duality ILNPv6 UDP typical run sequence numbers at the MN.	198
F.13	Enlarged plots of mobility-multihoming duality MIPv6 RTP typical run throughput for ToS.	200

F.14	Enlarged plot of mobility-multihoming duality MIPv6 RTP typical run sequence numbers for ToS.	201
F.15	Enlarged plots of mobility-multihoming duality ILNPv6 RTP typical run throughput for ToS.	202
F.16	Enlarged plot of mobility-multihoming duality ILNPv6 RTP typical run sequence numbers for ToS.	203
F.17	Enlarged plots of continuous mobility ILNPv6 RTP typical run throughput for BBB.	205
F.18	Enlarged plot of mobility-multihoming duality ILNPv6 UDP typical run sequence numbers for BBB.	206
F.19	Enlarged plots of continuous mobility ILNPv6 RTP typical run throughput for ToS.	207
F.20	Enlarged plot of mobility-multihoming duality ILNPv6 UDP typical run sequence numbers for ToS.	208
F.21	Enlarged MP-TCP throughput facet graph at the MN.	210
F.22	Enlarged MP-TCP throughput facet graph at the CN.	211
G.1	Datasheet of EdgeRouter X from Ubiquiti Networks website: https://dl.ubnt.com/datasheets/edgemax/EdgeRouter_X_DS.pdf	213
G.2	Datasheet of EdgeRouter 6P.	214
G.3	Datasheet of EdgeOS.	215

List of Tables

2.1	A table describing the use of names in IP. Based on a table in [1].	12
2.2	A table summarising different mobility and multihoming solutions.	26
3.1	A table showing the use of names in IP compared to ILNP: Unlike ILNP, IP-based naming uses the IP address across different protocol layers, leading to entanglement [1].	38
4.1	Testbed equipment information.	58
4.2	Lifetime and interval settings of router advertisement on access networks.	60
6.1	A table describing the new updated Locator Update codes currently in the draft RFC.	114
7.1	A table showing the properties of each footage. The average throughput was derived from 30 runs.	136
7.2	A table showing metrics used with comments illustrating the purpose of each metric.	138

7.3	Tables show PSNR results for ILNP (a) and MIPv6 (b) in continuous-mobility scenarios. Each cell contains a tuple of the following metrics: (1) the mean ratio of infinite PSNR frames (this value is what is used for the table colouring, as represented by the scale on the right), (2) the mean minimum PSNR value, (3) the difference between the mean maximum and minimum PSNR value, and (4) the number of runs with finite PSNR value frame/number of runs. The mean ratio of infinite PSNR frames indicates the mean ratio of <i>'perfect'</i> frames in each run, the higher the better. The mean minimum PSNR indicates the average <i>'worst case scenario'</i> , indicating the worst case quality metric on average. The difference between the mean maximum and minimum PSNR value indicates the average size of fluctuation in the video quality. (4) indicates how many runs out of the total number of runs were used in calculating the first three elements of the tuple. . .	154
7.4	A table showing the mean throughput for each clip in 0ms continuous-mobility scenarios. The average throughput is derived from 30 runs by dividing the total payload length per run received by the duration of the run. As MIPv6 hand-offs result in multiple-second loss of throughput, the overall throughput achieved is much lower than ILNPv6, as ILNPv6 provides smooth handoff with near-zero loss. The final column is a reference-only measurement without mobility hand-offs. ILNPv6, while providing continuous handoffs, achieves very similar overall throughput as the reference IPv6-only throughput.	157

- 7.5 A table showing PSNR results for the ILNPv6 mobility-multihoming duality scenario. Each cell contains a tuple of the following metrics: (1) the mean ratio of infinite PSNR frames (this value is what is used for the table colouring, as represented by the scale on the right), (2) the mean minimum PSNR value, (3) the difference between the mean maximum and minimum PSNR value, and (4) the number of runs with finite PSNR value frame/number of runs. The mean ratio of infinite PSNR frames indicates the mean ratio of *‘perfect’* frames in each run, the higher the better. The mean minimum PSNR indicates the average *‘worst case scenario’*, indicating the worst case quality metric on average. The difference between the mean maximum and minimum PSNR value indicates the average size of the fluctuation in the video quality. (4) indicates how many runs out of the total number of runs were used to calculate the first three elements of the tuple. 160
- 7.6 A table showing the mean throughput for each clip in 0ms mobility-multihoming duality scenarios. The average throughput is derived from 30 runs by dividing the total bits received in a run by the duration. The final column is a reference-only measurement without mobility handoffs. ILNPv6, while providing continuous multihomed soft-handoffs, achieves very similar overall throughput as the reference IPv6-only throughput. 160

List of Publications

- [1] R. Yanagida and S. N. Bhatti, “Seamless Internet connectivity for ubiquitous communication,” in *PURBA2019, Pervasive Urban Applications Workshop (UBICOMP 2019)*, Sep 2019, p. 12. [Online]. Available: <https://doi.org/10.1145/3341162.3349315>
- [2] R. Yanagida and S. N. Bhatti, “End-to-end networking with ILNP in Linux,” in *Netdev 0x13, Technical Conference on Linux Networking*, Mar 2019, p. 11. [Online]. Available: <https://netdevconf.org/0x13/session.html?talk-ilnp>
- [3] S. N. Bhatti, R. Yanagida, K. Shezhad, and D. Phoomikiattisak, “ilnp-public-1,” Sep. 2019, first software release of ILNPv6 prototype implementation containing mobility mechanism. [Online]. Available: <https://ilnp.github.io/ilnp-public-1/>

Chapter 1

Introduction

Today's computing environments rely on connectivity. Common modern applications rely on access to the Internet. Since the beginning of the Internet, various connectivities have emerged, providing more agility and capacity — along with devices that are more capable and mobile than ever before. However, fundamentally, the way of using connectivity is unchanged. The key issue here is that movements of hosts across different networks (network host mobility) and utilising multiple connectivities available to the host (network host multihoming) are still not possible today with typical end-systems. This thesis will present how mobility and multihoming duality can be realised by providing a way to migrate and utilise them simultaneously. This work shows that these two functions can indeed be a duality and can be realised as a duality function.

1.1 Host mobility

Host mobility is a feature where an end-host with multiple connectivities moves from one point of attachment to another. In the current Internet, many devices are mobile, such as smartphones, tablets, and laptops. Because of the entanglement of names in the current Internet Protocol (IP) networking architecture due to the overuse of IP addresses, such a feature is not possible without one or more of the following:

1. Middlebox
2. Application-specific mechanism
3. Network (provider) support

Middleboxes pose several issues; one of them is the introduction of a single point of failure (SPoF). This makes the mechanism more fragile and potentially introduces a security or privacy threat for man-in-the-middle (MITM) attacks. Application-specific mechanisms pose challenges in terms of deployment and uptake as legacy applications will be left behind, especially if the mechanism prevents the older version from functioning with the new version. Network support requirements will limit the network that the host can potentially ‘move’ to and from, narrowing the scope of the mechanism and deployment. There have been various attempts to provide host mobility for the Internet as a general solution, but none of them are deployed widely.

1.2 Host multihoming

Host multihoming enables the use of multiple connectivities available to a host simultaneously and continuously. This has several applications, such as failover, resilience to denial of service (DoS) traffic attack, and load-sharing. Specifically for mobile devices, host multihoming may improve the throughput using load-sharing, as mobile connectivity is often constrained.

Today, host multihoming is not well supported with IPv4, and with IPv6, the support is present but limited. Similar to host mobility, it often needs a middlebox, application-specific mechanisms, and/or support from network infrastructures, with the respective caveats.

1.3 Host mobility-multihoming duality

Host mobility-multihoming duality is when both mobility and multihoming are occurring simultaneously. A host with mobility-multihoming duality capability will be able to add or remove network connectivities on the fly, and will be able to make use of more than one of them actively for both sending and receiving. Such a mechanism can enable a much more dynamic use of network connectivities, especially in the mobile context. For example, in situations where a mobile host is moving through an area with public Wi-Fi networks, it can increase the available bandwidth dynamically by making use of multiple connectivities along with its existing cellular connectivity. If the public Wi-Fi networks happen to be congested, unlike a host with the current single-homed mechanism, it will not be ‘*stuck*’ with the public Wi-Fi and will still be able to utilise the cellular connectivity; once the public Wi-Fi becomes less congested, the host will be able to benefit from the additional bandwidth.

Currently, there are no known general solutions to mobility-multihoming duality for the Internet Protocols. More specifically, there are no solutions at the network layer which allow different transport protocols to function, and to function over different layer 2 protocols. There are proposals that enable mobility or multihoming individually, but it is not clear whether they can work as a duality.

1.4 Importance of the network layer, end-host-only solution

As mentioned, many proposed solutions enabling mobility and multihoming individually require changes beyond the network layer. However, any changes either below or above the network layer introduce unnecessary complications and narrow the scope of the solution.

Solutions that need coordination at layer 2 require distinct mechanisms that enable inter-workings between each *pair* of layer 2 technologies, and within the technology itself, if that does not exist already, meaning that up to n^2 mechanisms are required. Even then, the solutions involving layer 2 cannot provide mobility or multipath transport on their own — they still require layer 3 (network layer) support. Figure 1.1 (a) illustrates the complexity of providing support

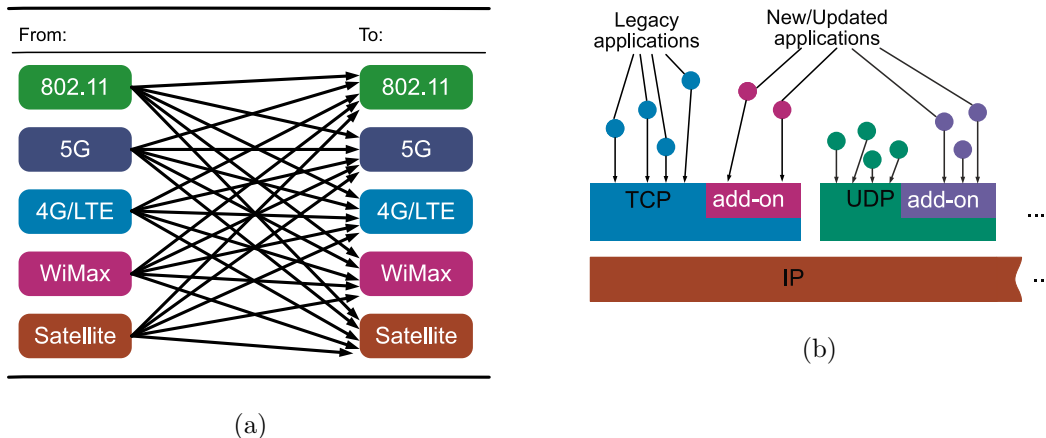


Figure 1.1: Diagrams illustrating the issues of providing additional network functionalities at the layers above and under the network layer (layer 3). (a) illustrates the complexity in enabling interworkings individually at layer 2. The arrows illustrate the distinct solution for enabling mobility from and to the respective connectivities. (b) illustrates the narrow scopes of mechanisms built into the transport layer. Any mechanism at the transport layer is only applicable to applications using the specific transport layer protocol. Even with the ‘same’ transport protocol — i.e. TCP vs MP-TCP — any additional feature often requires a modified application to make use of it.

across different technologies. As shown, since each technology is distinct, a handoff from one to another requires distinct solutions for each destination layer 2 technology and for each source layer 2 technology. In addition, within the specific technology, a handoff from one communication cell to another may need to be developed if it does not already exist. Figure 1.1 (b) illustrates how solutions above layer 3 can only cater for a very narrow scope. Solutions requiring coordination or changes at layer 4 and above will require solutions independently, fragmenting the scope and increasing the effort required overall. As these solutions reside further up above the network layer, the scope that they can serve is further limited.

Challenges in utilising multiple connectivities across different physical media have been a problem of interest. Some layer 2 technologies have a handover mechanism. However, those handover mechanisms do not cover handoffs across different technologies and they do not coordinate. Institute of Electrical and Electronics Engineers (IEEE) 802.21 standards [5, 6, 7] have been developed to tackle this problem, but they have not been proved or widely deployed. As IP is implemented at layer 3, this requires cross-layer coordination. IP mobility itself is still required separately at layer 3 as Media Independent Service (MIS) as described in the 802.21 framework.

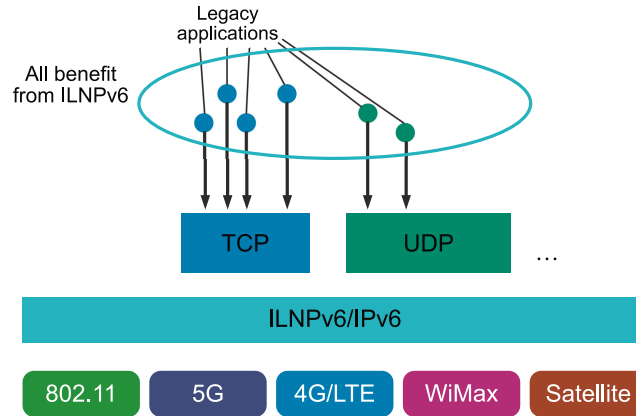


Figure 1.2: A diagram illustrating the wide scope of the impact from the solution at the network layer. A solution implemented at the network layer enables the inter-workings of different technologies at the lower layers, while still providing functionalities to different transport layer protocols and all applications making use of these transport layer protocols.

Above the network layer, solutions such as MP-TCP, Stream Control Transmission Protocol (SCTP), and QUIC exist as approaches for mobility and multihoming. However, because those solutions are only applicable to specific transport layer protocols, legacy applications cannot use those solutions, narrowing the scope. Hence, it is not feasible as a general solution.

At the network layer, MIPv6 does exist to provide mobility. However, MIPv6 requires additional infrastructure, which poses challenges in deployment.

In summary, as illustrated in Figure 1.2, the network layer is the correct layer to enable mobility and multihoming, as it provides inter-workings for lower technologies, provides a general solution for a wide range of applications, and the solution can be free of additional infrastructure.

1.5 Thesis statement

Thesis statement: *Mobility and multihoming duality can be realised through an end-host solution, without requiring changes to core infrastructures or existing applications.*

The above thesis statement can be broken down into several requirements.

A. Mobility and multihoming duality:

A host can utilise multiple connectivities for both sending and receiving simultaneously, while being able to dynamically change the connectivities used.

B. An end-host solution:

Hosts participating in the communication are the only part of the system that have to change, that is:

1. Without requiring changes to core infrastructures:

No changes to network devices, such as routers, and no additional proxies or middleboxes are needed.

2. Without requiring changes to applications:

No changes to applications (existing application programming interfaces (APIs) must be maintained).

1.6 Thesis structure

The structure of the thesis is as follows.

This chapter introduced the problem, challenges, the thesis statement, and the thesis structure.

Chapter 2 discusses the fundamental issues and challenges, historic approaches to the naming issue, and solutions to mobility and multihoming individually. This chapter also presents current approaches to mobility and multihoming, comparing and contrasting the proposed solutions and their issues.

Chapter 3 introduces Identifier Locator Network Protocol (ILNP), the architecture, and ILNPv6 as the engineering solution to ILNP. This chapter presents a consolidated summary and comparison against existing mobility and multihoming mechanisms.

Chapter 4 presents the problem space, testbeds, scenarios, and common metrics used across the three sets of evaluations, as well as the engineering contributions of the thesis.

Chapter 5 presents how ILNPv6 is extended to enable continuous mobility. The continuous mobility scenario is evaluated, comparing ILNPv6 and MIPv6 using iperf2 TCP and UDP flows.

Chapter 6 presents how mobility-multihoming duality is achieved with ILNPv6. The chapter presents how ILNPv6 in Linux kernel is extended to enable mobility-multihoming duality as an end-host-only solution, functioning in the testbed presented earlier. Empirical evaluation is conducted using iperf2 TCP and UDP flows.

Chapter 7 examines application level performance in both continuous mobility and mobility-multihoming duality scenarios, examining the performance using Quality of Experience (QoE) metrics evaluating real-time video clips sent and received over the testbed network. The continuous mobility scenario evaluation compares results from ILNPv6 and MIPv6. Similar to Chapter 6, application level performance is evaluated in the mobility-multihoming duality scenario.

Chapter 8 summarises the thesis, contributions, discussion, and future work.

Chapter 2

Background

Over the years, the Internet has changed dramatically; specifically, use cases and devices connected to the Internet have greatly changed and expanded. The numbers of applications, devices, and users of the Internet have grown more than ever before. As a result, the diversity of devices that connect to the Internet and the ways they communicate over the Internet have increased dramatically. One of the notable changes to the Internet is the rise of mobile devices, which are capable of connecting to the Internet using various technologies over distinct network technologies, often moving dynamically across network boundaries. Many devices are physically capable of using multiple connectivities; if an appropriate host multihoming solution existed, they could potentially benefit from the additional connectivities, improving the performance of the data communication. As the name suggests, mobile devices switch connectivity from one network to another as they move through physical space, which, without an appropriate mobility mechanism, results in disruptions to ongoing communication. A user may think that since their device has multiple physical networks available most of the time, it should utilise them accordingly by switching between them as it moves across different network boundaries, or even utilise them simultaneously, without any interruptions to the data communication and operation of applications using the Internet. However, these intuitively desirable features are still not possible with typical mobile devices. In the context of host mobility, current mechanisms are either limited to a specific layer 2 technology and its specific domain, limited to a specific application, and/or

require a cross-layer solution needing significant changes to the core infrastructures. On the other hand, host multihoming is often limited to applications that are specifically modified to achieve this with a specific transport layer protocol designed for multipath transport. This is due to a fundamental issue with naming in the current Internet, specifically, the entanglement in the roles attached to the names used in network protocols. As mentioned, there are solutions to achieve either mobility or multihoming to a certain extent, but none of the current solutions achieve both at the same time. Solutions that claim to enable both have not shown whether they function simultaneously or not. This chapter will introduce the naming problem in the Internet Protocols preventing mobility-multihoming duality and present historic identifier-locator split architecture approaches to tackle the issue. The chapter will also analyse existing approaches to enable mobility and multihoming.

2.1 History of IP networking and previous attempts to enable identifier-locator split

2.1.1 Internet Protocol and its naming architecture

The ARPANET, which established the technical basis of the current Internet, was the first wide area network to implement the Internet Protocol suite, also known as Transmission Control Protocol (TCP)/Internet Protocol (IP) (TCP/IP). The Internet was built on top of what was discovered in the process of developing the ARPANET. As one of the contributions of the ARPANET, TCP/IP became the ‘de facto’ standard protocol suite on the Internet. As the Internet evolved, one of the side effects that became apparent was the overuse of IP addresses in the protocols above the network layer. In the earlier phase of the Internet, domain name system (DNS) did not exist. Without the existence of DNS, users needed to use the IP addresses directly in the applications unless the addresses were already defined statically in the configuration files, such as `/etc/hosts`. The TCP socket application programming interface (API) design came from the 4.2 BSD, which needs IP address to identify a host. Such circumstances have contributed to how addresses were directly used in the programs that used the BSD socket APIs and also across the different protocol layers above the network layer. As the Internet grew, more

protocols appeared on top of the network layer. IP addresses made more appearance as the key host identifier throughout the layers above the inter-networking layer. A number of protocols used the IP address as the sole identity of the host. TCP and User Datagram Protocol (UDP) sockets lookup had to be done using IP addresses, the packet checksum was calculated using the source and the destination addresses, and some applications identified different hosts using the IP addresses, assuming a host would only have a single IP address. It is true that the use of IP addresses was, at times, the only way, but it was purely out of convenience in the other cases. The consequence of this use of IP addresses across a wide range of layers was that the IP address not only encapsulated one of the identities of the host, which was bound to the interface, and was also topologically significant at the same time. It not only identified the host with a globally unique identity, but also provided a topological location of the host within the Internet. This semantic overloading of the IP addresses created entanglement in naming in the protocol suite. Because the IP address serves both roles, any changes in the device's point of attachment changed its identity across different protocols above the network layer. When the point of attachment of the host changes, since the identity of the host used in protocols at the network layer and above changes, any sessions bound to the host will reset and the applications will need to re-initialise their communication. Such implications render mobility practically impossible without complex mechanisms to mitigate them. The primary cause is the end-system protocol state — TCP and UDP sockets have a tuple containing the source address, the destination address, the source port number, and the destination port number. Changes to the host's IP address will disrupt the end-system protocol state, disrupting the connection. In addition, checksum is done using the tuple, which will also be invalid if any part of the tuple changes.

An early approach to mobility was Mobile IPv4 (MIPv4). MIPv4 is an IETF mobility standard for IPv4, which requires a proxy device, the Home Agent (HA), to manage the device's topological locations and carry out tunnelling. This requires HAs not only at the host's 'home network' but also at the 'foreign network', called Foreign Agent (FA). Without them, the host cannot perform a handoff to the foreign network and would not be reachable via its 'home address', which topologically belongs to the host's 'home network'. Such limitations renders MIPv4 impractical to as a general solution.

As the Internet grew, the development of the next generation Internet Protocol to replace

IPv4 had began with the IPng working group, which started in October 1994 [8]. The IPng later became IPv6 with the RFC1883 [9] published in December 1995. IPv6 introduced a larger address space and a unicast address format with slightly clearer separation between the network part and the identity; with its 128-bit address space, the unicast address format splits the address into a 64-bit network routing prefix and a 64-bit interface identifier (IID). However, the previously mentioned entanglement still remains, as IPv6 continues to use full the 128-bit to identify the host at upper layers, and as the name IID suggests, the lower 64-bit value is specific to a certain interface, binding the upper-layer identity to it.

2.1.2 Known issues in the naming architecture in the Internet —

The entanglement of the name spaces

IP packet forwarding was designed to be simple; the design focused on delivering the packets, providing a *general solution* to a wide range of applications, leaving any application-specific mechanisms to protocols residing above the network layer. Routers forward IP packets by looking up the destination IP address in the header and selecting the most specific route available until the packets reach their destinations; the addresses are topologically significant, therefore allowing routers to make decisions to deliver the packets. At the same time, once the host receives them, the transport layer protocols use IP addresses to distinguish the different correspondents that the hosts are communicating with — IP addresses are used as the identifier of the hosts. Since the IP address represents both the topological location of the host and its identity — tying the host’s location to its identity — it is a rigid and inflexible name in the current Internet architecture. Any changes to the IP address during a communication session would cause disruption to protocol states that are using the IP addresses. In addition, because it contains the topological location, it is tied to the specific network that the interface is attached to.

The idea that IP addresses currently serve too many purposes and that they prevent flexible use of network connectivity is not particularly new. Mark Weiser stated in 1993 that the current IP is “insufficient” for mobility, as “higher-level protocols such as TCP assume that underlying names will not change during the life of a connection” [10]. He used the analogy of a block-number being used as the filename to illustrate how harmful it is to overload name and topology onto

a single value. Brian Carpenter later presented a similar argument in his article “IP addresses considered harmful” published in 2014 [11]. The article highlights operational issues caused by the current usage of IP addresses. This includes the inability to refer one host to another, difficulties in dual-stack operations, challenges in multihoming and managing multiple interfaces, problems in roaming, and disruptions to communications during the renumbering of a site. The article also analysed different approaches to replace the IP address.

Both works discuss the issue of using the IP address above the layer it belongs to, which binds the states of the upper layers tightly and results in inflexibility. Table 2.1 summarises the issues: the overuse of the IP address across different layers and how the address is statically bound to the interface, therefore binding the upper layer protocols to a specific interface.

Protocol Layer	IP (IPv4 and IPv6)
Application	FQDN / IP address / Application
Transport	IP address
Network	IP address
(interface)	IP address

Application: Application-specific naming
 FQDN: Fully Qualified Domain Name

Table 2.1: A table describing the use of names in IP. Based on a table in [1].

2.2 Historic Identifier-Locator split architectures

As described in previous sections, the current IP architecture suffers from entanglement in naming across the layers. To solve this problem, a number of efforts suggested identifier-locator split architectures in the past.

2.2.1 8+8/GSE draft by Mike O’Dell

Mike O’Dell proposed identifier-locator split naming with IPv6 in his 8+8 Internet-draft [12] in order to reduce the route-scaling problem. Although the main intention was not to enable

mobility nor multihoming, identifier-locator split naming solves the entanglement, which is a step towards enabling them.

The 8+8 draft was written in 1996, not long after RFC1883 [9], which is one of the earlier IPv6 specifications published as a proposed standard. The 8+8 draft later developed into GSE, “Global, Site and End-System Designator”, which the IPng working group analysed in 1999. The working group concluded in the draft “Separating Identifiers and Locators in Addresses” that without an “accompanying authentication system”, the complete separation of the identifier and the locator “renders the identifier untrustworthy”, expressing concerns in separating identifiers and locators [13].

8+8/GSE’s identifier-locator names are as follows:

End System Designator (ESD): A globally unique 8-byte-long name that every entity in the network should have, a flexible name that is not topologically significant

Routing Goop (RG): An 8-byte-long name for determining topological location in the global Internet

The main concept of 8+8 is simply splitting the name into a topologically significant part and a topologically insignificant part. Such naming architecture would require the upper layers to only consider ESDs when performing operations on the headers, such as TCP checksum and socket lookups. Another core concept of 8+8 is the separation between “public topology” and “private topology”. While current IPv4 networking consists of “private network” and “public network” and their corresponding address types, this is not part of the design, but rather a consequence of having to aggregate many ‘private’ nodes into one public global address. 8+8’s end-to-end ESD integrity and its topologically insignificant aspect allow implementation of these network types: the site-border-routers can perform the conversion between ‘global’ RG and ‘private/internal’ RG. While it has not been deployed, it was one of the early examples of an identifier-locator split architecture over the IPv6 address space.

2.2.2 Nimrod architecture

Nimrod is a map-encap identifier-locator split routing architecture proposed to tackle a growing number of more diverse hosts. Nimrod describes a group of networks as graphs, whereby a node

represents a network, and an edge refers to the link between two networks. The edge is described by a pair of bits referring to the two connected entities. Nimrod has also not been deployed, but its naming architectures and concepts describe how an identifier-locator split can realise advanced networking features, such as mobility and multihoming. The architecture is defined in RFC1992 (INFORMATIONAL) [14]. The names used in Nimrod are as follows:

End-System Identifier (EID): The identifier in the identifier-locator split architecture; it does not have topological significance.

Endpoint Label: Human readable ASCII strings referring to an EID, similar to a fully qualified domain name (FQDN). A comment in RFC1992 [14] suggests that an “Endpoint Label” will be needed in reality, much like how an FQDN is used to refer to a host.

Locator: The locator is assigned to a ‘node’, which in Nimrod, describes topological entities of the network, such as the site, the area, etc. “Adjacency” is a pair of “nodes” describing the topological relations between them. An end-host can have multiple locators while a node can only have one locator.

Nimrod routers contain a map, which is a graph representing the topology using nodes and adjacency. The nodes in this map contain a locator, and routing decisions are based on locators, not EIDs.

2.3 Mobility options across different technologies

As mentioned in Chapter 1, there are solutions to enable host mobility within respective layer 2 technologies. This section describes how they relate to this work, specifically with layer 3 solutions like Identifier Locator Network Protocol (ILNP). In general, any new technology requires a new handoff mechanism, if the handoff is done at layer 2. As shown in Figures 1.1 and 1.2, while some solutions that exist at layer 2 enable handoff across different layer 2 technologies, as more technologies emerge, more handoff mechanisms must be developed. Even if the handoff is implemented, if the handoff is attempted across the two distinct layer 3 domains with distinct subnets, a layer 2 solution itself is not sufficient.

2.3.1 Cellular technologies

Cellular technologies such as 5G and 4G have their own mobility solutions; since the base station retains control over how the individual stations, i.e. the cellular devices, operate, and the gateway to the Internet is further upstream in the network, mobility can be achieved within its network. The current cellular technologies do enable 3G to and from 4G handoffs and 4G to and from 5G handoffs. However, as discussed, these require definitions of two separate mechanisms or some adjustment if they are at all shared. Moreover, upper layer mobility solutions are still needed when: (1) the host moves or connects to and prioritises another connectivity available nearby (e.g. IEEE 802.11 Wireless-Fidelity (Wi-Fi)), and (2) the host crosses the boundaries of the mobile networks' IP subnets (which may happen especially with cross-generation i.e. 4G to/from 5G), experiences handoffs, and/or roams to another carrier (e.g. some Mobile Virtual Network Operators (MVNOs) use several physical cellular networks).

In very limited cases for limited applications, a technology known as Wifi-Calling is available to provide mobile phone service using the nearby IEEE 802.11 Wi-Fi network that is not provided by the carrier. This is a very specific vertical handoff mechanism only applicable to a specific application; it enables mobile devices to access the core networks to receive and make calls via IEEE 802.11 networks when the device has no cellular connectivity.

2.3.2 IEEE 802.11

The IEEE 802.11 family contains 802.11/r and other roaming mechanisms. Naturally, these need to occur within the subnet, and must have the same service set identifier (SSID).

Some implementations of IEEE 802.11 Wi-Fi network involve sending the whole IEEE 802.11 frame to a central controller across a large site, which can unify the subnet and provide a consistent IP address regardless of the geographical and topological location of the device. However, this is also limited to a specific network and hardware combination, and the handoff has to be within the network.

In addition, none of the roaming mechanisms in the IEEE 802.11 family will support vertical handoff to/from ethernet or cellular connectivities, requiring higher layer mobility solutions.

2.4 Multihoming options at the lower layer

It is not possible to provide host multihoming at a layer lower than layer 3. Host multihoming refers to a functionality which allows a host to utilise multiple *networks*, i.e. multiple distinct IP subnets; that is, a host connects to multiple distinct layer 3 networks, such as connecting to multiple Internet service providers (ISPs). This is not possible at all without involving layer 3 mechanisms. Features to utilise multiple interfaces, such as link aggregation, exist to increase the capacity, but this is not host multihoming. Link aggregation is typically only used to bundle multiple interfaces within the same layer 2 domain, therefore it is within the same subnet, against just one other layer 2 node. These bundled interfaces are presented as a single virtual interface with a single IP address. However, host multihoming requires layer 3 solutions.

2.5 Current and previous solutions that approach mobility and multihoming separately

The following sections discuss existing solutions for enabling mobility and multihoming respectively.

Generally, the following approaches are used:

Transport layer approach: Modifications to the transport layer at the end-host.

Shim layer approach: Introduction of a shim layer between the network layer and the transport layer at the end-host.

Network-based approach: Introduction of new network entities or changes to the network infrastructure itself.

Transport layer approach Solutions at the transport layer can integrate well with the application, which can enable good Quality of Service (QoS) as the transport layer protocols have the potential to adapt their behaviour to cater for a certain application. However, simultaneously, a transport layer solution is only available to the specific transport layer protocol, requiring a separate solution and deployment effort for each individual transport layer protocol. Moreover,

they are either an add-on to the existing transport layer protocol or an entirely new protocol, which requires applications to change, limiting their benefits to applications that are updated or specifically built to take advantage of the solution.

Shim layer approach Shim layer approach solutions allow identifier-locator split by associating multiple IP addresses and translating them before interacting with transport layer protocols. The shim layer mechanism resides between the transport layer and network layer. It functions by translating locator addresses to and from the primary address, acting as the identifier, before passing the packets to and from the upper layer protocols. While this can provide an end-host-only solution, such approaches break end-to-end continuity. In addition, this approach segments the role of IP addresses; some IP addresses become an identifier to a host, while others only act as a locator to a host. This approach does not fix the entanglement, but rather worsens it by creating two new uses of IP addresses, further overloading their semantics.

Network-based approach Network-based approach solutions require changes in the core infrastructures, such as routers or additional middleboxes. In general, such mechanisms, when deployed, have the potential to provide additional features to many hosts at once, as they do not always require changes to the end-hosts. However, they are often complicated, requiring significant changes to the infrastructure, which may be expensive both in terms of financial cost, as well as engineering effort to implement and maintain them. In addition, if the solution does not involve any changes to the end-host, the additional functionalities will only work within specific networks that implement the solution, limiting the availability.

The following sections introduce respective solutions to mobility and multihoming, categorised in the different types of approaches described in this section.

2.6 Mobility solutions

Mobility solutions enable data communication to continue despite the device's change of attachment to the Internet. The following describes existing approaches to IP mobility.

2.6.1 Shim layer approach

Dynamic Internet Mobility for End-Systems (DIME)

Dynamic Internet Mobility for End-Systems (DIME) [15] is a host-based shim layer approach mobility solution. Similar to Level 3 Multihoming Shim Protocol for IPv6 (SHIM6), it provides a shim layer that identifies a socket with a primary IP address acting as the identifier and translates to the current IP address as the locator. DIME uses a signalling protocol called Internet Host Mobility Protocol (IHMP) to notify its locators to the correspondent node. The identifier is called Host Identifier (HID), which is kept in the *Socket Table* that resides in the DIME. The *Socket Table* simply keeps track of sockets, the current address that the sockets are bound to, and the HID they are associated with. In addition, the HID table is used to keep track of HID, its ‘active’ and ‘unreachable’ foreign IP addresses as locators, and its own current local IP address that is reachable from the foreign HID. IHMP packets are encapsulated in UDP packets. DIME uses the IP address directly as the HID, which avoids introducing additional namespaces to refer to the hosts. The translation, much like SHIM6, occurs between the transport layer and the network layer. On Linux systems, DIME is implemented as a loadable kernel module (LKM), eliminating the need to build a custom kernel. While it is an end-host-only solution, as it is a shim layer approach, the typical drawbacks of shim layer approaches are present, such as the fragmentation of the name space and disruption to the architecture.

2.6.2 Network solutions

Mobile IPv6 (MIPv6) and its variants

Mobile IPv6 (MIPv6) is the IPv6 variant of Mobile IP (MIP), the successor of MIPv4 [16, 17]. MIPv6 is defined in RFC6275 [18] in proposed standards status. Plain MIPv6 is a part end-host, part network solution. It requires a mobility management server, the HA, to track the Mobile Node (MN)’s topological location and proxy the packets to and from the Correspondent Node (CN). Meanwhile, MN needs a user-land daemon to establish a tunnel to the mobility management server. Unlike MIPv4, the MN does not require a FA to be present in the network it migrates to. However, the requirement for HA remains and therefore, the presence of *home*

network still remains, along with the use of the two IP addresses to refer to the MN. MIPv6 uses two types of addresses for MN: Home Address (HoA) and Care-of Address (CoA). CoA is the locator, as it indicates the current topological location, while HoA is the identifier for referring to the host globally; and the packets are delivered to CoA by either a proxy through the MN's HA or directly from the CN if Route Optimisation (RO) is available (RO enables the CN to be informed about the CoA). While the MN can move to a network without an HA, additional infrastructure at the home network, on top of the host requiring modification, poses challenges in deployment. In addition, if the MN or the CN cannot enable RO, it requires sub-optimal triangular routing via the MN's HA. This adds unnecessary traffic load to the home network both in terms of ingress and egress, potentially adding latency and a throughput bottleneck between the MN and the CN. Furthermore, the more MNs a home network has to support, the more the home network and the HA suffer from the increasing load for tunnelling traffic for the MNs'.

MIPv6 variants include: Fast-handover for Mobile IPv6 (FMIPv6) [19], which improves handover performance; Proxy Mobile IPv6 (PMIPv6) [20], which is a fully network-based mobility management solution eliminating the need for changes in end-hosts; and Hierarchical Mobile IPv6 (HMIPv6) [21], which introduces hierarchy into mobile management to increase its handoff performance.

Proxy Mobile IPv6 (PMIPv6) PMIPv6 is a fully network-based variant of the MIPv6, defined in RFC5213 [20]. Unlike the MIPv6, with its mobility management signalling coming from the end-host itself, PMIPv6 relies on network components to carry out signalling, management and tunnelling of the packets. PMIPv6 introduces Mobility Access Gateway (MAG) and Local Mobility Anchor (LMA). An LMA acts similar to an HA; it allocates the HoA and manages traffic coming in and out of PMIPv6 networks. Within the network managed by a MAG, the MN receives an address that is mobile, the HoA. The address of a MAG is the CoA, which the LMA uses to forward the packets to the MN. As the MN moves from one MAG to another, the MAGs notify the LMA of the MN's change in the point of attachment. The MAG and the LMA signal each other to confirm the MN's movement and set up tunnels and routes. The LMA will be the gateway to the rest of the Internet, while the MAG will provide the tunnel to

the LMA. Unlike MIPv6, it does not require end-host modification, the infrastructure is a lot more complex, and it limits the MN's movement only to specific networks that have PMIPv6 infrastructures. Limitations common across different proxy-based solutions apply to PMIPv6, such as the performance bottleneck, additional attack vectors, and the introduction of single points of failure (SPoFs).

Network Mobility (NEMO) and Mobile Ad-hoc Network Mobility (MANEMO) Network Mobility (NEMO), defined in RFC3963 [22] is an approach to mobility where an entire network moves its point of attachment to the Internet. This enables mobility to a group of hosts under the network, allowing hosts without mobility mechanisms to move across different points of attachment.

Mobile Ad-hoc Network Mobility (MANEMO) is a mechanism enabling network mobility combined with the mobile ad-hoc network (MANET) mechanism, allowing mobile networks to have separate egress and ingress paths [23].

These mobile network solutions require significant work in order to be deployed, increasing their cost to implement and to maintain.

Distributed Mobility Management (DMM)

Distributed Mobility Management (DMM) is an approach to enable mobility management in distributed architecture, decentralising the mechanism. The Internet Engineering Task Force (IETF) DMM-working group (WG) [24] aims to specify solutions that enable such mechanisms by extending the existing mobility standards, such as MIPv6 and its variants. DMM aims to enable a more scalable mobility mechanism by a decentralised approach. However, it still suffers from the complexity of the mechanism posing challenges in deployment along with respective caveats of the solutions that DMM proposals involve.

2.6.3 Transport layer solution

QUIC

QUIC is a UDP-based transport protocol, which became a proposed standard in RFC9000 [25] in May 2021. It was initially developed by Google as part of Hyper Text Transfer Protocol (HTTP) version 3 (HTTP/3), later developed by the IETF working group, QUIC-WG, to the proposed standard. QUIC is positioned as a new transport layer network protocol; however, it is built on top of UDP, using UDP as a wrapper to encapsulate QUIC packets into IP packets and pass them to user-land applications via the standard UDP socket interface, avoiding implementing the protocol in the operating system (OS). RFC9000 [25] defines mobility solution as ‘connection migration’. The primary application of QUIC is HTTP/3, which is deployed as part of browser software updates. As it is a transport layer solution and a *new* transport layer protocol, applications must change to use QUIC, which does not allow legacy applications to benefit from it. In addition, QUIC still uses IP addresses, but introduces “Connection identifier”, a new namespace only accessible within QUIC.

2.7 Multihoming solutions

A common use case of multiple connectivity is multihoming, where a host utilises multiple connectivities available to increase its bandwidth or resilience to failures. The following sections describe approaches to enable multihoming, including both site and host multihoming.

2.7.1 Shim layer approach

Level 3 Multihoming Shim Protocol for IPv6 (SHIM6)

SHIM6 [26] is an end-host shim layer approach multihoming solution. SHIM6 introduces the initial IP address as an Upper-layer Identifier (ULID) and the set of subsequent IP addresses as Locators (Ls). ULID is the identifier, while L is the locator. SHIM6 functions by providing a translation layer between the transport layer and the inter-networking layer, translating the ULID to and from Ls. The host will initiate a transport layer session using regular address selection mechanisms and session initialisation mechanisms. RFC5533 [26] recommends the use

of heuristics in the implementation to decide when to pay the overhead before commencing its operation.

The main goal is to enable host and site multihoming, load-sharing and failover. The setup phase of multihoming is a 4-way exchange, where it establishes the ‘contexts’ — a name introduced to describe pairs of ULID and L. When the nodes involved need to change the set of Ls, an Update Request message is used. The host that received an Update Request will reply with an Update Request Acknowledgement. Although this enables multihoming while being transparent from the transport layer, the aforementioned caveats of shim layer still exist.

2.7.2 Network solutions

Locator Identifier Separation Protocol (LISP)

Locator Identifier Separation Protocol (LISP), defined in RFC6830 [27], proposes a network-based solution to realise identifier-locator split. It is a map-and-encap solution. LISP introduces EID and Routing Locator (RLOC): EID is topologically insignificant, while RLOC is topologically significant. They both re-purpose IP addresses to represent respective names. LISP is comprised of the following components: Ingress Tunnel Router (ITR), Egress Tunnel Router (ETR), Proxy Ingress Tunnel Router (PITR), and Proxy Egress Tunnel Router (PETR). ITRs are located at LISP enabled locations, encapsulating outgoing packets of the site into LISP packets using EID to RLOC mapping before being sent out to the Internet to reach the other LISP site(s). An ETR receives LISP packets for the site that it resides in, and removes LISP encapsulation before forwarding the decapsulated packets to the destination end-system within the site. PITR and PETR are proxy tunnelling routers to serve for non-LISP sites to forward to LISP sites. A LISP router is a device that can do all or some of the tunnel router functionalities. Since this is a network solution that requires significant modification to the site-network, the deployment involves changes to routers and the benefit of LISP is limited to networks with LISP implemented.

Multihomed with MultiPrefix (MHMP)

RFC7157 [28] titled “IPv6 Multihoming without network address translation (NAT)”, describes issues and some potential solutions in order to enable Multihomed IPv6 hosts and networks without the use of IPv6 Network Prefix Translation (NPTv6). The document defines Multihomed with MultiPrefix (MHMP) hosts, which are IPv6 hosts multihomed with multiple links; these hosts with multiple global IPv6 prefixes do not utilise NPTv6. More specifically, the document discusses three scenarios: a host in a site with two routers on a single interface, a host in a site with single site border router with two uplinks, and a host with two uplinks. The biggest challenge in utilising multiple links with the absence of NPTv6 is the fact that the source address selection, next hop selection, and DNS server selection are either incorrect or non-deterministic when a host has access to multiple connectivities either through the site-border-router or at the host itself. The Request For Comments (RFC) suggests using Dynamic Host Configuration Protocol (DHCP)v6 options to distribute policy to manage those configurations. In all three scenarios presented in the RFC, none of them are possible without a further modification to the host. Furthermore, this does not enable multipath-transport, but rather, allow access to multiple paths where a socket can only utilise one of them at a time.

2.7.3 Transport layer solutions

MultiPath TCP (MP-TCP)

MultiPath-TCP (MP-TCP) [29, 30] is a transport layer solution to allow multi-path transport. An MP-TCP socket consists of sub-flows with different pairs of addresses/ports, while presenting a single TCP socket to the application. While MP-TCP presents TCP socket API to the application, additional options can expose MP-TCP specific information. Both parties need to be MP-TCP capable. MP-TCP capable hosts will initiate a three-way handshake with MP-TCP specific options; `MP_CAPABLE` is used for setting up a MP-TCP capable main flow and `MP_JOIN` is used to add a subflow. As the name suggests, it is a solution specific to TCP, so it will be challenging, if not impossible for UDP applications to adapt to MP-TCP. In addition, this also segments the name space where some IP addresses act as locators, and some act as identifiers.

Stream Control Transmission Protocol (SCTP)

Stream Control Transmission Protocol (SCTP) [31] is a transport layer protocol which enables multi-path transport. SCTP, similar to TCP, is connection oriented but comes with greater flexibility. In particular, while allowing loss-prevention, strict-order checking can be disabled. While the main goal was multihoming support by allowing multiple addresses to bind to a connection, mobility can be achieved by dynamically adding and removing the addresses of the mobile node. However, for both parties to be mobile, an additional mechanism, provided by a ‘Cooperation Server’, maintaining binding addresses of MNs, is needed. SCTP communications are described as ‘association’ and allow both stream and message oriented data transmission. The association is initialised using a four-way handshake, which generates nonce for off-path attack prevention. During the initialisation, addresses available from both parties are exchanged and heartbeat messages are used to detect the reachability of each address. RFC5062 [32] points out possible security threats of SCTP, such as denial of service (DoS) attacks and man-in-the-middle (MITM) attacks. While MP-TCP has a way for legacy applications to use it via OS kernel, SCTP presents an entirely different socket API, different from TCP or UDP. Therefore, the application must change greatly to benefit from SCTP’s features.

2.8 Current approaches to mobility-multihoming

There are several solutions proposed to enable host mobility and multihoming for the Internet. They can be classified into following three categories: end-host solutions, network-based solutions and transport layer solutions. The following sections introduce the existing solutions and analyse the pros and cons of the existing solutions.

2.8.1 Comparison criteria

The following sections discuss current solutions for mobility and multihoming. In order to compare and analyse, the following criteria will be used.

Naming

How names are used

Layer

Which layer it operates on

Mobility

Whether the solution supports mobility or not as part of its initial design

Multihoming

Whether the solution supports multihoming or not as part of its initial design

Change in end-systems

Whether the solution requires the end-host OS to be modified or not

Change in applications

Whether the solution requires changes in the application itself or not

Requires network support

Whether the solution requires core network infrastructures to support it or not

Require additional entities

Whether the solution requires any additional network entities or not

Table 2.2 summarises the solutions enabling mobility and multihoming. As shown, Identifier Locator Network Protocol v6 (ILNPv6) is the only solution that has considered both mobility and multihoming from its initial design. In addition, ILNPv6 is also the only solution that enables both mobility and multihoming while operating solely at layer 3 (the network layer) as an end-host-only solution, without introducing an additional shim layer, changes to applications, changes to core network infrastructures, or requiring any additional network entities.

Criteria	LISP	MIPv6	SHIM6	DIME	HIP	MP-TCP	QUIC	ILNPv6
Naming Layer	EID&RLOC L3	HoA&CoA L3	ULID&L Shim + L3	HID&IP Shim + L3	HI&IP Shim + L3	IP L4	IP L4	NID+L64 L3
Mobility	A	I	A	I	A	A	I	I
Multihoming	I	U	I	A	A	I	A	I
Change in end-systems	No	Yes	Yes	Yes	Yes	No	No	Yes
Change in applications	No	No	No	No	Yes	Yes	Yes	No
Requires network support	Yes	No	No	No	No	No	No	No
Requires additional entities	Yes	Yes	Yes	No	Yes	No	No	No

Keys:

EID End-System Identifier **CoA** Care Address **HI** Host Identifier **Shim** a shim layer between L3&L4
RLOC Routing locator **ULID** Upper-layer Identifier
HoA Home Address **L** Locator **NID** Node Identifier

For the Mobility and Multihoming criteria:

A — **Add-on** indicates that the functionality was not included in the initial design. Instead, the functionality was added separately.

I — **Initial Design** indicates that the functionality was part of the initial design.

U — **Unsupported** indicates that the functionality is not available.

Table 2.2: A table summarising different mobility and multihoming solutions.

2.9 Host mobility and multihoming solutions

The following are solutions that enable host mobility and host multihoming.

2.9.1 Network solution

LISP and LISP-MN LISP, defined in RFC6830 [27], is a network-based-solution to realise identifier-locator split. Locator Identifier Separation Protocol Mobile Node (LISP-MN) is a variant of LISP that supports mobile node outside of LISP networks, first drafted in 2016 [33]. The mobility management is done using a Map Server, which could be learnt from a Map-Resolver. LISP-MN is an active Internet-Draft which is still a work-in-progress in September 2021, titled draft-ietf-lisp-mn-10 [34]. Due to its complex signalling during the handoff, the disruption time is long — about 3 seconds of no throughput even with the improved version of LISP-MN [35]. While both mobility and multihoming are defined, it is not clear whether they work simultaneously.

2.9.2 End-host based shim layer approach

SHIM6 SHIM6 [26] is an end-host based shim layer approach multihoming solution. While it is built for multihoming, by dynamically adding and removing the Ls, host mobility can be realised. Dhraief et al. proposed how that could be done [36]. Rahman et al. also proposed a mobility mechanism with SHIM6 called SEMO6, SEamless MObility using SHIM6 [37]. However, those proposals do not solve the naming issue, and it is not clear whether mobility and multihoming mechanisms function simultaneously or not. In addition, at the time of writing, there is no public implementation of SEMO6.

DIME DIME [15] is another end-host based shim layer approach to mobility described in Section 2.6.1. While the current version only supports mobility, [15] proposes that the Stack-Trans can be further developed to support multihoming. However, there are no full proposal or implementation available.

2.9.3 End-host based re-naming approach

Host Identity Protocol (HIP) Host Identity Protocol (HIP) [38, 39] is a strictly end-host based network protocol which uses cryptographic keys as host identifiers. HIP was not designed initially to enable mobility and multihoming; rather, the primary aim was to provide more secure communication between hosts. However, as it introduces identifier-locator split, later documents proposed mobility [40] and multihoming [41] mechanisms that make use of HIP respectively, which later became part of its architectural standard documentation in July 2021 [42].

HIP introduces Host Identifier (HI), a cryptographic public key representing the Identity of a host; and it also introduces Host Identifier Tag (HIT), an operational representation of HI that is only 128 bits, much shorter in length than current commonly used public key, which are 2048 bits or 4096 bits. In HIP, the use of IP addresses still remains; the IP address is used as a locator, and the full 128 bits of the IPv6 addresses bound to each interface are used. The HIP host initiating communication, called Initiator, starts HIP Base Exchange (BEX) by sending a message to the peer, called Responder. This message initiates a handshake, with the purpose to exchange Diffie-Hellman keys and to generate the session-key. Once the handshake is complete, a HIP association, a shared state between two HIP hosts, is created. The binding of an HI to an FQDN is done using the DNS. As the IP address no longer serves any part in host identification, the socket has to be re-written for HIP, requiring new API for applications [43]. As it is a strictly host-based solution, networking components such as routers will not be able to participate, thus site-multihoming and mobility cannot be achieved, nor is traffic engineering based on HIT possible. To enable host mobility, a rendezvous mechanism is needed [44], which involves additional network infrastructure/entities.

Meanwhile, a fully network version of HIP exists. The HIP Proxy, similar to PMIPv6, is designed to avoid changes to end-hosts. Thus, it suffers from the general problems of proxy-based solutions, as well as the problem of being restricted to a specific topological domain.

2.9.4 Transport layer approach

MP-TCP As described in Section 2.7.3, MP-TCP is a transport layer multihoming-first solution. Although its original goal was multi-path transport or multihoming, by adding and

removing subflows during transmission, it can be used for mobility [45]. There are proposals of integrating mobility management to enable mobility for MP-TCP [46]. However, there are no formal standards or implementations present at the time of writing.

QUIC As described in Section 2.6.3, QUIC only has mobility as part of the core standard. There is a proposal to add multihoming, which is in the draft status as of March 2021 [47]. However, it is also unclear how the mobility and multihoming will function together.

2.10 Current available solutions

The following are solutions that are currently available and can be tested.

Existing host mobility solution — MIPv6 Mobile IPv6 (MIPv6) [18] is an IETF standard solution to IPv6 mobility. However, it is still not widely adopted due to its deployment and management complexity. Unlike ILNPv6, MIPv6 is a mobility-only solution that requires an additional network entity, the HA. It does not have multihoming capability either.

Existing host multihoming solution — MP-TCP MultiPath-TCP (MP-TCP) is defined in RFC8684 [30]. MP-TCP operates by initialising main-flow, then sub-flows are added to the session. While MP-TCP itself is not designed to support mobility, mobility extension is being developed for MP-TCP. Unlike ILNPv6, as it is a transport-layer solution, MP-TCP requires applications to change in order to enable multihoming.

Emerging ‘transport’ protocol — QUIC QUIC is a UDP based transport protocol, which became a proposed standardised status in May 2021 by the IETF working group, QUIC-WG in RFC9000 [25]. QUIC-WG has defined mobility for QUIC and some of the implementations include mobility. Multihoming is proposed in draft status as of September 2021. However, similar to MP-TCP, QUIC requires QUIC-capable applications and is unable to provide mobility or multihoming to legacy applications.

2.11 Qualitative analysis of MP-TCP

MP-TCP is a multihoming solution at the transport layer. The implementation is present for Linux kernel as a custom kernel that can be installed. To provide more thorough comparisons between ILNPv6 and MP-TCP, the following presents *qualitative* analyses of MP-TCP in the same movement scenario as presented earlier in this chapter.

It is not possible to conduct a complete direct comparison for the following reasons:

1. MP-TCP is a TCP only solution — comparison can only be made with TCP scenarios with ILNPv6.
2. MP-TCP is a transport layer solution, while ILNPv6 operates at the network layer.
3. The MP-TCP implementation available did not allow for the scheduler to change — MP-TCP's default scheduler was used, while ILNPv6 used the deficit round-robin (DRR) scheduler.
4. It does not implement mobility-multihoming duality — mobility is not the focus of the design.

Therefore, the comparison is limited to *qualitative* analysis and for TCP flow only. To conduct the comparison, the testbed presented in Section 4.4 was used, with the scenario presented in Section 4.5.2. The scenario is the same as the mobility-multihoming duality evaluation conducted using ILNPv6 in Chapter 6. To conduct the *qualitative* analysis, MP-TCP was setup as outlined below:

Operating System: Debian Buster was used, as its default kernel is version 4.19 — the same as the latest 0.95.2 MP-TCP kernel is based on

MP-TCP kernel: Verion 0.95.2 (available from github)¹

Path-manager: Set to 'fullmesh'

Scheduler: Set to 'default'

This is the default scheduler setting recommended. This selects path based on round trip time (RTT) and congestion-window.

¹<https://github.com/multipath-tcp/mptcp/releases/tag/v0.95.2>

Routing: As required, a routing table was created for each interface

Existing scripts available on <https://multipath-tcp.org> did not support IPv6 — A new script was written to achieve the same routing table configuration.

iproute2: A custom `iproute2` was installed to disable an interface from being used by MP-TCP.

The script was updated to make use of this, similar to how ILNPv6 has a `sysctl` configuration path to disable an interface and enable make-before-break.

iperf2: The same version of `iperf2` program used for the previous ILNPv6 evaluation was used.

The same scenario to the one described in Section 6.3.1 was used. The MN moved from single connectivity to three-way connectivity gradually, then return back to single connectivity.

2.11.1 Results

Similar to ILNPv6, the result of 20 runs were collected. Both the median and mean throughput were **10 Mbps**. However, due to the scheduler and its behaviour, the throughput distribution was very different.

Figure 2.1 on page 32 and Figure 2.2 on page 33 show the throughput plot over the 120 seconds run. They show the throughput received at the MN and the CN respectively. As shown, the behaviour of the scheduler leads to a ‘bursty’ behaviour of throughput changes. Although the aggregate is consistent, the ‘bursty’ behaviour of the throughput could cause issues in the traffic management of the networks that the respective interfaces are connected to.

2.11.2 Discussion

One of the issues with MP-TCP is potential security threats. As described by Popat and Kapadia in [48], MP-TCP has several security threats, such as session hijacking. Therefore, the current application of MP-TCP has been often limited to known good servers with known good clients, such as Apple’s mobile platforms’ notification delivery [49].

In addition, as shown in the results in Section 2.11.1, for MP-TCP, with its specially designed TCP congestion control algorithm, the host and the networks exhibited very bursty loads, without

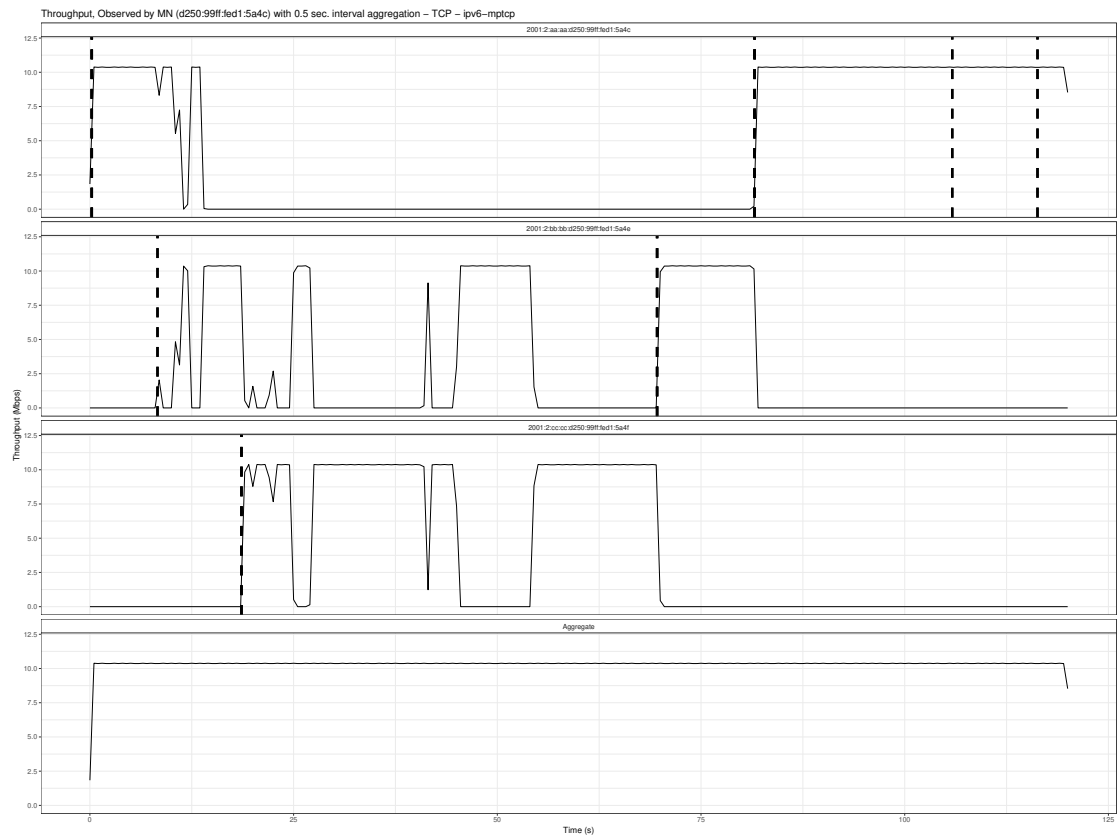


Figure 2.1: Throughput facet plot of MP-TCP flow received at the MN. The top three plots show the throughputs received at the addresses of the respective three interfaces at the MN and the bottom plot shows the aggregate throughput. The distribution of the throughput is uneven and changes to the throughput on the individual interfaces are ‘bursty’. The vertical line shows the protocol level signalling (here, MP-TCP specific multihoming management signal) to add or remove a connectivity received at the respective addresses.

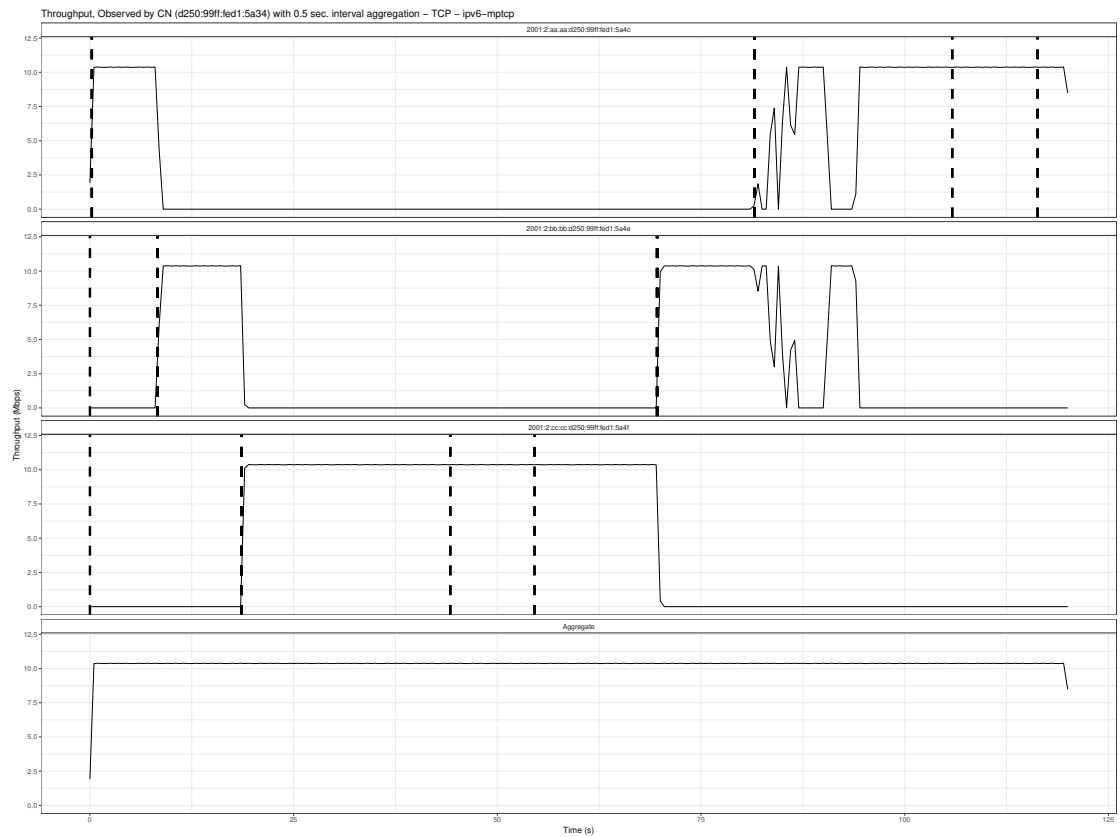


Figure 2.2: Throughput facet plot of MP-TCP flow received at the CN. The top three plots show the throughputs received from the addresses of the respective three interfaces at the MN and the bottom plot shows the aggregate throughput. The distribution of the throughput is uneven and changes to the throughput on the individual interfaces are ‘bursty’. The vertical line shows the protocol level signalling (here, MP-TCP specific multihoming management signal) to add or remove a connectivity received from the respective addresses.

the load being anywhere near the limit of the paths or the host. Its data transmission characteristic is not as smooth and consistent as ILNPv6 as shown in Chapter 6. Such behaviours are not ideal, and can lead to further congestions on the network.

2.11.3 Summary

MP-TCP was designed with different objectives to ILNP. The primary focus of MP-TCP is *resource pooling*, where the focus is to making the best use of available bandwidth instead of enabling mobility-multihoming duality. Coupled with security issues and ‘bursty’ behaviour, MP-TCP, while available, is not widely used. Therefore, there is still space for an alternative solution, which is presented in Chapter 6.

2.12 End-to-end principle and ILNP

The previous sections showcased several solutions to enable mobility and multihoming. First of all, none of them proposed *mobility-multihoming duality* — rather, the two features are distinct from each other in the solutions without any suggestions or definitions as to how they function together. More crucially, all of the solutions, except for HIP, do not propose changes in the naming. Designs that do not address the naming issue have several issues, one of them being the disruption to the end-to-end state between the hosts. Disruption to end-to-end states causes several issues, such as additional overhead and disruption to the architecture.

While HIP proposes changes in naming, it still utilises IP addresses — HIP enables identifier-locator split by inserting the HIP layer between the transport and the network layers. The insertion of an additional layer in the stack disrupts architectural design, preventing any networking components from interacting with HIP traffic. More crucially, this method of renaming involves API changes, requiring applications to be modified in order to function.

With ILNP, the solution enables *mobility-multihoming duality* from the initial design at the network layer. This is enabled by its renaming approach, which solves the fundamental issue — the entanglement of names in the IP networking stack. By solving the entanglement, and by purely operating at the network layer, ILNP enables mobility and multihoming without

disrupting the architecture, while maintaining the end-to-end principle, and providing those functionalities to legacy applications.

2.13 Networked applications on the Internet

One of the biggest changes to the Internet is its usage and the type of traffic it carries. As more devices are capable of processing larger amounts of data, more multimedia traffic has become commonplace in the Internet. In the early days of the Internet, MIDI files and still images were the primary multimedia payload. Nowadays, high-definition video transmission is a common application of the Internet.

The Cisco Annual Internet Report (2018–2023) [50] specifically mentions following trends:

- Increase in mobile devices and connectivity reaching more people
- Increase in diversity of devices connected to the Internet
- Increase in bandwidth capacity and usage on various connectivity such as cellular, fixed broadband, and Wi-Fi
- Increase in use of videos and video devices connected to the Internet

With the rise of mobile devices and popularity of video calls, real-time videos seem to be a type of payload that is becoming increasingly common. In fact, modern TV broadcast systems utilise IP [51]².

However, providing real-time-video for mobile devices poses a challenge. As described, the point of attachment changes the identity, disrupting the transport layer state tuple, requiring applications to reset and re-establish the connection, potentially requiring user-intervention to do so. While each application may implement a mobility handling mechanism, it requires a significant effort in address management, and implementing path selection mechanism is far from trivial. In fact, the route selection is generally carried out inside the operating system, making it more challenging to enable multipath connections for soft-handoffs. To truly enable ubiquitous real-time video applications for the Internet, a smooth, continuous handoff is crucial.

²For example, the British Broadcasting Company (BBC) showcases how live television broadcast is distributed over the Internet.

<https://www.bbc.co.uk/rd/projects/dynamic-adaptive-streaming-ip-multicast-dasm>

2.14 Summary

This chapter discussed the fundamental issues and challenges in enabling flexible connectivity. The use of names has been an issue as early as the 1990s. This issue not only prevents connectivities from being used flexibly, it also became the barrier to true ubiquitous communication and thus, ubiquitous computing. There have been many approaches to tackle the naming issue, mobility, or multihoming. However, many require additional middleboxes, application-specific mechanisms, or network support. In addition, none of them have been deployed widely. ILNP, on the other hand, tackles the fundamental barrier to dynamic communication by tackling the entanglement in the name space as an end-host-only solution that does not require network support. The following chapter introduces ILNP and its engineering solution ILNPv6.

Chapter 3

Introduction to Identifier Locator Network Protocol

The previous chapter described existing solutions to mobility and multihoming and determined that Identifier Locator Network Protocol v6 (ILNPv6) has the properties to enable mobility-multihoming duality. This chapter follows to introduce the novel ILNPv6 as the chosen approach in this thesis to realise *mobility-multihoming duality* at the network layer as an end-host-only solution.

3.1 Identifier Locator Network Protocol (ILNP) — Architecture

Identifier Locator Network Protocol (ILNP) employs identifier-locator split architecture, where the name referring to a node is distinct from the name referring to the location of the node. A Node Identifier (NID) is a name given to a ‘node’, rather than an interface. A node could be a virtual/logical node or a physical node. The node, in this context, is a physical device connected to a network using one or more interfaces, such as computers, phones, tablets, etc. A locator, on the other hand, is a name given to the network in which the node resides. The interface will use the locator given by the network from the border router dynamically. A pair of identifier and

locator is called Identifier-Locator Vector (I-LV). The end-hosts application uses fully qualified domain name (FQDN) or its own specific application naming mechanisms. Transport protocols identify hosts using a NID. The network protocol and devices utilise locators to forward packets through the network.

As described in RFC6740 [52], the core concept of ILNP lies in solving the entanglement in the namespace in today’s Internet.

Protocol Layer	IP (IPv4 and IPv6)	ILNP (ILNPv6)
Application	FQDN / IP address / Application	FQDN / Application
Transport	IP address	NID (dynamic binding to L64)
Network (interface)	IP address	L64 (dynamic binding to NID & interface) (dynamic binding to L64)
	Application: Application-specific naming FQDN: Fully Qualified Domain Name	L64: Locator (64 bits) NID: Node Identifier (64 bits)

Table 3.1: A table showing the use of names in IP compared to ILNP: Unlike ILNP, IP-based naming uses the IP address across different protocol layers, leading to entanglement [1].

Table 3.1 describes the different usage of names between the Internet Protocol (IP) and ILNP. As shown, with ILNP, transport protocols, such as User Datagram Protocol (UDP) and Transmission Control Protocol (TCP), are not bound to the network in which the host resides. In fact, in the traditional IP, some applications also use IP addresses directly. As ILNP allows dynamic bindings, transport protocols and applications are no longer tied to the interface, thus the identifier of the host is also independent of the network topological location.

3.1.1 Locator Update (LU)

Locator Update (LU) is the signalling mechanism that notifies changes to the use of locators between two ILNP hosts. LU is sent from a host to the Correspondent Node (CN) participating in communication. Once received at the CN, the changes are made to its local data structure, which holds information about the sender; the CN then replies with a Locator Update Acknowledgement (LU-Ack). As shown in Figure 3.1, the signalling mechanism is a lightweight 2-way signalling, so it only requires one round trip time (RTT) to update the CN’s information about the Mobile Node (MN).

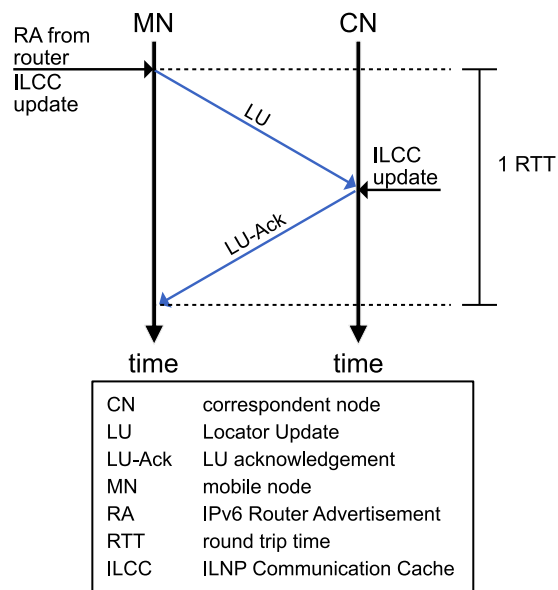


Figure 3.1: Timeline diagram of a Locator Update (LU) messages: In order for an MN to update its CN with its changes to network connectivity, an LU is sent; the CN then updates its ILCC accordingly, and then replies with a Locator Update Acknowledgement (LU-Ack). This only requires one RTT.

3.2 Identifier Locator Network Protocol v6 (ILNPv6) —

Engineering solution

As ILNP is an architecture, there could well be a clean-slate implementation of ILNP. Such a solution, while technically possible, would be rather unrealistic, given how backwards incompatibility between IPv4 and Internet Protocol (IP)v6 is disrupting IPv6 deployment. As IPv6 was decided as the ‘next-generation IP’ by the Internet Engineering Task Force (IETF) IPng working group [8], it is natural to seek an engineering solution for ILNP to function with IPv6. In fact, not only should the engineering solution of ILNP coexist with IPv6, but also function with legacy IPv6-ready-applications. Thus, one of the focus in implementing such a solution is to co-exist and allow backwards compatibility with IPv6 and allow legacy ILNP-unaware applications to function on ILNP hosts. ILNPv6, the engineering solution of ILNP over IPv6, was designed and implemented as such. To enable such a design, ILNPv6 has a ‘wire-image’ [53] of IPv6. In other words, as soon as the packets leave the host, they appear as conventional IPv6 packets to network devices such as routers. To implement ILNPv6, much of the existing IPv6 implementation is reused, allowing ILNPv6 to be enabled by setting a run-time parameter in the operating system (OS) kernel¹ and allowing the communication to fallback to IPv6. As application programming interfaces (APIs) remain unchanged, legacy applications, as long as they use a name to refer to a host, can utilise ILNPv6. ILNPv6 is defined in a series of Internet Request For Comments (RFC) documents in the Internet Research Task Force (IRTF) under RFC6740–RFC6744 [52, 54, 55, 3, 56]; they are currently in *Experimental* status.

3.2.1 NID and L64 in IPv6 ‘wire-image’

The global IPv6 unicast address is a 128-bit address, where the topological location and the identity of the network interface are already partly separated except for some specific cases [57]. More specifically, the first 64 bits are the global network prefix, while the last 64 bits represent the interface identifier (IID). The IID is unique to each network interface, and it can be generated from the layer 2 media access control (MAC) address. ILNPv6 utilises this split that already exists to encode the Locator (L64) and the NID in the IPv6 ‘wire-image’ as described in RFC6741

¹sysctl parameter in Linux systems

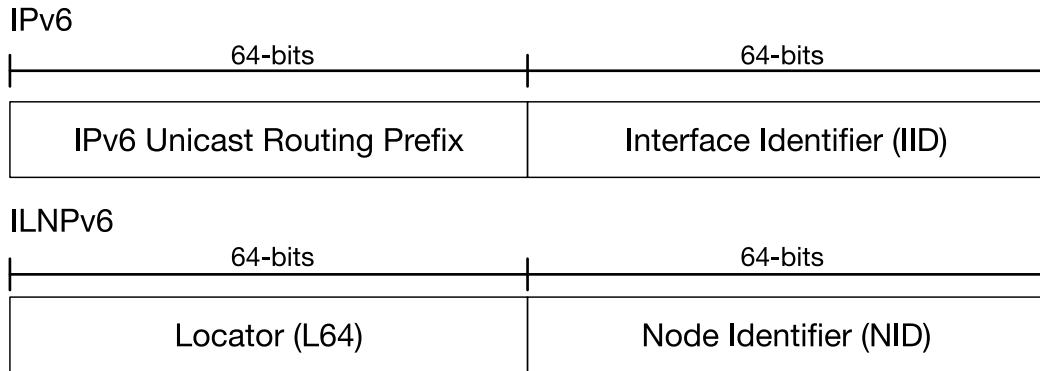


Figure 3.2: A diagram showing address/IL-V formats and the sizes of each part for the respective network protocols [1]. ILNPv6 enables identifier-locator split using the already existing prefix-IID split of the IPv6 unicast addressing.

Section 3.1 [54], L64 in place of the global network prefix, and NID in place of the IID as shown in Figures 3.2 and 3.3. Hence, on an ILNPv6 node, all network interfaces present the same NID. Due to the use of the IPv6 ‘wire-image’, ILNPv6 achieves backwards compatibility both in terms of the networking infrastructure as well as applications. As long as an application creates sockets via the C socket APIs and refers to another host using names — regardless of whether the name is resolved by the `/etc/hosts` file or using domain name system (DNS) — if the entry is an ILNP host entry, the application does not need to change to utilise host-wide ILNPv6 features, such as multihoming and mobility. In addition, implementing ILNPv6 can be achieved by reusing much of the existing packet processing mechanism for IPv6, and by modifying parts of it just for ILNPv6 packets.

3.3 Previous work on Identifier Locator Network Protocol v6 (ILNPv6)

In the past, the following works have been done to realise mobility and multihoming individually by leveraging the identifier-locator split architecture of ILNP.

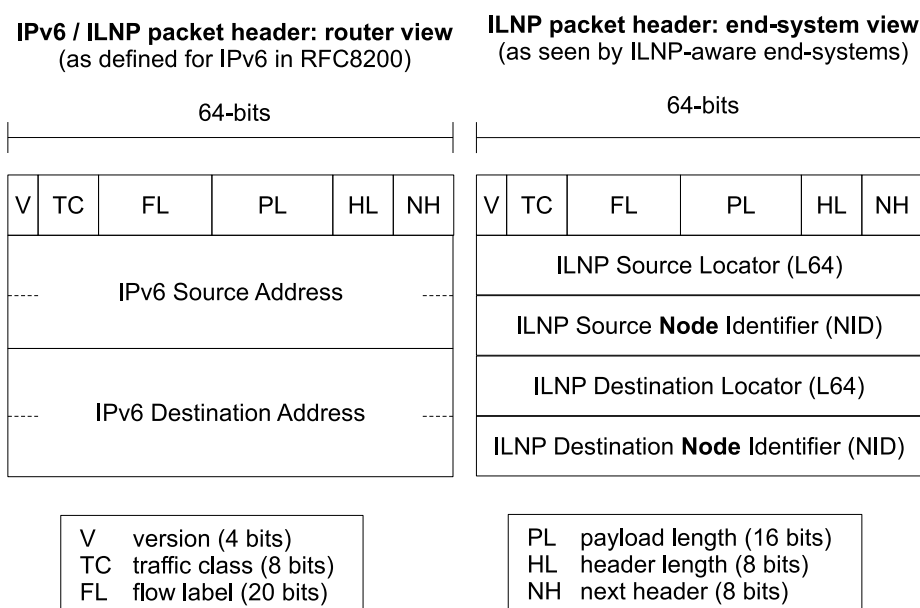


Figure 3.3: A diagram comparing packet headers for IPv6 and ILNPv6 packets [1]. The ‘wire-image’ of ILNPv6 packets, in terms of address, as shown, lines up with the IPv6 global unicast prefix and IID. For ILNPv6, a nonce follows, which is added using the Destination Option Header, as defined in Section 4.6 of RFC8200 [2].

3.3.1 IP host mobility with ILNPv6 on Linux kernel 3.9

Previous work by Ditchaphong Phoomikiattisak [58] on mobility showed a prototype implementation of ILNPv6 on Linux kernel version 3.9. It showed ILNPv6 hard- and soft-handoffs where a host moves from one network to another in one direction, and once within a single communication session. The comparative study with in-depth protocol level analysis showed the advantage of ILNPv6 over Mobile IPv6 (MIPv6) in terms of its simplicity in signalling, performance, and smaller overall protocol overhead. However, it was limited to a single handoff during the communication session and only focused on a single handoff performance analysis.

3.3.2 IP site multihoming with ILNPv6 on FreeBSD

Previous work on ILNPv6 implementation on FreeBSD done by Bruce Simpson [59] showed IP multihoming. It demonstrated site- and host-multihoming [60], with evaluations using iPerf UDP synthetic flow and Internet Control Message Protocol v6 (ICMPv6) packets. The primary focus of the work was on enabling site-multihoming with ILNPv6. Therefore, the work showed

multihoming with two site-border-routers to evaluate network resilience, an application of site-multihoming. The implementation provided resilience against a potential loss of connectivity by utilising ILNPv6 site-multihoming and an additional signalling mechanism to coordinate the two connectivities between the two side-border-routers. While the work presented host multihoming functionality, it only enabled two-way load sharing across the two networks where both hosts reside.

Both works evaluated the respective functionalities of ILNPv6 by using synthetic payloads generated by benchmarking softwares. While both mobility and multihoming mechanism exists for ILNPv6, they are independent solutions from one another, implemented on two separate operating systems.

3.4 ILNPv6 on Linux kernel 4.9.52

Previous work on ILNP by Ditchaphong Phoomikiattisak [58] was done using Linux kernel version 3.9. The experiments presented in the following chapters of the thesis use ILNPv6 implemented on Linux kernel version 4.9.52, based on the aforementioned implementation on Linux kernel version 3.9. Linux kernel version 4.9 was selected as it was the latest long-term support (LTS) version of the Linux kernel in 2017.

ILNPv6 implementation in the Linux kernel contains two new files: `net/ipv6/ilnp6.c` and `include/net/ilnp6.h`, providing core ILNPv6 data structure and functions.

`ilnp6.c` and `ilnp6.h` are the core part of ILNPv6. ILNPv6 hosts maintain two sets of NID and L64 information, one for the host itself and one for corresponding nodes. The combination of NIDs and L64s are stored in the ILCC, which is defined and managed within the two files.

In addition to the two files added to the Linux kernel, existing files in the Linux kernel have been modified. Specifically, files implementing transport protocols, address configuration, and socket data structure have been modified to allow ILNPv6 to coexist with IPv6. Section 6.2.1 in Chapter 6 contains the list of existing files modified in Linux kernel as well as the modifications to transport-layer processing code.

3.4.1 Locator Update (LU) with ILNPv6

The signalling mechanism for ILNP is called LU, which is described in Section 3.1. In the context of ILNPv6, LU is implemented as a new ICMPv6 [61] message, as described in RFC6743 [3]. This is a known packet format that is supported on a standard IPv6 network. Therefore, the implementation on Linux kernel has also reused the existing ICMPv6 processing mechanism to enable LU packet processing.

3.4.2 Interface configuration for ILNPv6 hosts

Configuration of interfaces on ILNPv6 hosts reuses much of the same mechanism already present in IPv6. An ILNPv6 host can configure its NID and L64 using stateless address autoconfiguration (SLAAC) and router advertisement (RA). The host receives the RA, which configures the L64 on the interface, as an IPv6 host would configure the prefix portion of the IPv6 address on an interface. The difference between an IPv6 host and an ILNPv6 host is the way they configure the IID/NID portion respectively. IID is configured in several ways depending on the configuration of the host, but this is tied to the specific interfaces on the host. NID, on the other hand, can be configured dynamically to more than one interface on a host. After these are initially set, an ILNPv6 host will still use duplicate address detection (DAD) just like an IPv6 host.

3.4.3 ILNP Communication Cache (ILCC)

ILNP Communication Cache (ILCC) holds Identifier-Locator Vectors (I-LVs) for both local and remote hosts. For each NID, L64 entries are linked. Each L64 holds three attributes: preference, lifetime, and state. Preference may be used to decide which L64 is used in initiating communication. The lifetime field dictates how long the given L64 is valid for. For local L64, RA is used. For remote L64, the lifetime field in LU is used. The state field holds the current state of L64².

Figure 3.4 on page 46 describes the transition of the L64 states and the definition of each state. With ILNPv6 hosts, SLAAC [62] can be used to configure the interface. When a new interface is enabled, layer-2 link-up occurs, and then an RA is received. The RA contains the new IPv6 prefix, which is the new L64. The ILNPv6 host configures its NID in place of IID on

²The definitions of AGED and EXPIRED differ slightly from that in RFC6741 [54]; however, this is how it is implemented, with the occlusion of the AGED state. See Sections 5.1.4 and 5.5.7 for further details.

the interface. When a new L64 is received, a local ILCC L64 entry is added, with its initial state set to VALID. The L64 state will transition as follows:

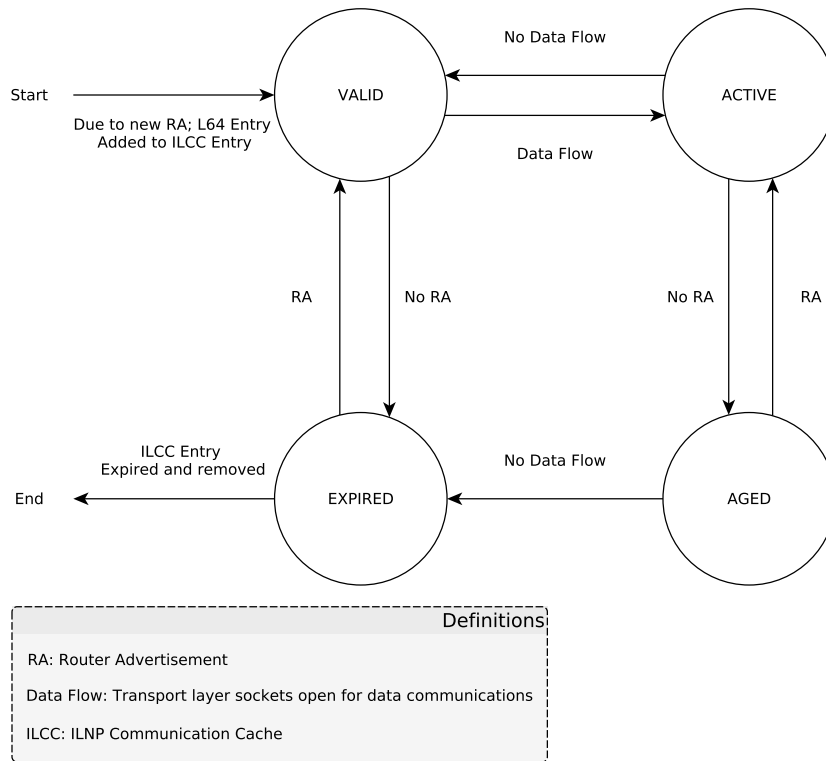
1. VALID as the initial state when the L64 is received via an RA
2. Data flow begins on the L64:
 - (a) The L64 becomes ACTIVE
 - (b) Once data flow ends, the L64 returns to VALID
3. The RA is lost and the L64 lifetime runs out:
 - (a) The L64 becomes EXPIRED
 - (b) If 2a has occurred prior, the L64 becomes AGED
 - (c) If 3b has occurred and then data flow ends, the L64 becomes EXPIRED
4. The RA is received after 3a, the L64 returns to VALID

3.4.4 ILNPv6 Nonce

ILNP has a built-in lightweight off-path attack protection called the Nonce [56]. In ILNPv6, this is implemented in the packets inside the IPv6 Destination Option header. This is used in the ILCC and in this thesis, and the nonce mechanism is not modified.

3.5 Consolidated summary and comparison against existing mobility and multihoming mechanisms

MIPv6 and MultiPath-TCP (MP-TCP) are the IETF standards-track mobility and multihoming solution respectively. However MIPv6 is not adopted widely and MP-TCP's application is currently limited to very specific domains as described in Section 2.11.



VALID: Locator within its lifetime, with no active data flow

ACTIVE: Locator within its lifetime, with active data flow

AGED: Locator beyond its lifetime, with active data flow

EXPIRED: Locator beyond its lifetime, with no active data flow

‘No Data Flow’: Transport layer session(s) using the locator has ended

‘Data Flow’: Transport layer session(s) using the locator has began; the locator is in active use

‘RA’: Receiving router advertisement with a prefix used for the locator

‘No RA’: The router advertisement with a prefix used for the locator is not received beyond prefix’s lifetime

Figure 3.4: The L64 state diagram describing how the L64 state transitions. The lists below the state diagram define the states and actions.

Mobile IPv6 (MIPv6) In the context of mobility, MIPv6, an official IETF mobility solution for IPv6, exists at layer-3. Similar to ILNPv6, since it operates at layer 3, transport protocols at layer 4 could be supported and it implements identifier-locator split naming to some extent. However, MIPv6 does not enable multihoming, and is a much more complicated system, both from the amount of infrastructure and signalling required.

The primary difference between ILNPv6 and MIPv6 is the use of naming and control plane design. While ILNPv6 re-designs the naming architecture, MIPv6 simply reuse and translate IP addresses. As outlined in Section 2.12, this breaks the end-to-end addressing. As described in Chapter 2, MIPv6 uses the Home Address (HoA), an address that is assigned at the home network as the node identifier, which is used at the transport layer to identify the host. When the host moves, the Care-of Address (CoA), the address at the foreign network, is used to send the packet to the host at the foreign network, which is used to send packets to and from Home Agent (HA) or the CN (if both hosts support Route Optimisation (RO)), which is then translated to HoA before being passed to the transport protocols. Due to this addressing model, MIPv6 requires HA, a middle-box that must be managed. For a CN to communicate with an MN, without RO, packets must be sent to the HoA, then sent to the CoA by the HA. Meanwhile, ILNPv6 only needs the initial I-LV and direct signalling, LU and LU-Ack to initiate handoff, whereas MIPv6 requires Binding Update (BU), Home Test (HoT), Care-of Test (CoT) to the HA, as well as the CN. More specifically, the HA must relay the Home Test Init (HoTI)/HoT and Care-of Test Init (CoTI)/CoT packets between the MN and CN. Compared to ILNPv6, MIPv6 requires at least six-fold signalling messages between the three hosts — MN, CN, and HA.

While MIPv6 is implemented as a kernel-module for Linux systems, it still requires user-land daemon `mip6d`. This must be configured on top of maintaining the HA at the home network. Meanwhile, ILNPv6 does not require any user-land daemon for basic operations and can be configured via the `sysctl` interface.

In short, while MIPv6 provides mobility at layer 3, it cannot enable multihoming, and it requires additional infrastructure and userland software — HA and `mip6d` respectively — further complicating the implementation and signalling.

Multipath TCP (MP-TCP) MP-TCP’s addressing model is similar to that of MIPv6; MP-TCP also uses the full IP address to create subflows using additional IP addresses, overloading the semantics of IP addresses. It also induces additional TCP states as each subflow is distinct.

While MP-TCP operates in-kernel, as shown in Section 2.11, it requires a significant amount of configurations. First, similar to ILNPv6, a custom kernel with MP-TCP implementation needs to be installed. Second, routing tables must be configured for each interface. This is either done statically or requires the if-up/if-down script, which only currently works for IPv4 and requires re-writing to support IPv6. Third, to disable MP-TCP, either customised `iproute2` program must be used, or flags `0x80000` must be set in `/sys/class/net/<INTERFACE>/flags` manually.

While many applications use TCP, UDP applications are becoming increasingly important with increased use of Voice over IP (VoIP) and other realtime applications as discussed in Section 2.13. As outlined in Section 2.11.2, MP-TCP’s application is rather limited. MP-TCP’s security issues are limiting its adoption. In addition, MP-TCP, with the purpose-built scheduler, still exhibits undesirable ‘bursty’ data transmission characteristics across multiple paths even without reaching anywhere near the limit of the respective paths as shown in Figures 2.1 and 2.2.

ILNPv6 operates at layer 3, transparent from layer 4. ILNPv6 uses new id-loc split architecture while still using the existing IPv6 infrastructure, eliminating the need for an additional full IP address pair for each additional path. ILNP’s design provides the potential to support any layer 4 protocols and enable mobility-multihoming duality, while using a clean end-to-end architecture.

3.6 Summary

This chapter introduced ILNP and its engineering solution, ILNPv6. ILNP presents a clean architecture to solve entanglements described in Chapter 2. ILNPv6 leverages the existing IPv6 infrastructures to introduce this clean naming architecture to the Internet as a superset of the IPv6. As described in both Chapters 2 and 3, ILNPv6 has the architectural potential and foundation for enabling mobility-multihoming duality as an end-host-only solution.

Currently, there are no widely-deployed mobility solutions for IP for end-hosts. There

are no widely-deployed host multihoming solutions either. In extension, there is no mobility-multihoming duality solution. Part A. of the requirements derived from thesis statement shown in Section 1.5, '*Mobility-multihoming duality*' is not possible today. At the moment, it is not clear whether mobility-multihoming duality is possible without specific network support as all proposed solutions, except for ILNPv6, rely on support from the network provider or an additional network entity; and ILNPv6 still implements them separately, on separate OSs. The following Chapter introduces the testbed and the engineering contribution required to evaluate ILNPv6's ability.

Chapter 4

Testbed and kernel development to enable empirical evaluation

In order to conduct empirical evaluations, physical testbeds were constructed. The evaluations conducted in the following chapters use the testbed described below with some adjustments as needed for each evaluation. This chapter also introduces terminologies and methodologies used across evaluations conducted in this thesis. The following section describes a simple mobility scenario to provide the context to the design of the testbed and empirical evaluation methods. In order to enable Identifier Locator Network Protocol v6 (ILNPv6) evaluation, significant engineering contribution was required. This chapter also outlines the engineering contribution that enabled the evaluations presented in the following chapters.

4.1 Problem space

As described in Chapter 2, mobile devices are becoming more popular as a type of device connected to the Internet. A common use case of mobile devices is mobile communication, such as Voice over IP (VoIP) or video calls. The following is an example of such a scenario, illustrated in Figure 4.1.

Suppose a user is about to leave their home with a smartphone which is connected to the

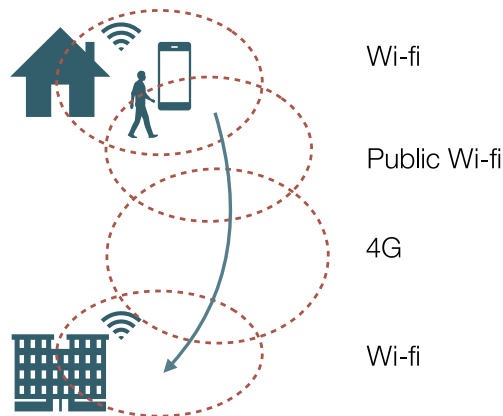


Figure 4.1: A diagram illustrating a potential scenario for the use case of a mobile device. A user begins a communication session at home via 802.11 Wi-Fi, passes through different network connectivities, and arrives at their place of work, which also provides 802.11 Wi-Fi.

home Institute of Electrical and Electronics Engineers (IEEE) 802.11 network (wireless LAN (WLAN), also referred to as Wi-Fi) and has cellular data connectivity available but not in use. They receive a VoIP call just before they leave. The call will begin over the 802.11 connectivity, as most smartphones are configured to make use of WLAN whenever possible to reduce the usage of often metered cellular connections. However, as they leave home, the smartphone will be disconnected from the home WLAN. Since there is a nearby public 802.11 network which the smartphone has associated with in the past, the device connects to the public 802.11 network. At some point, the user will then leave the public 802.11 network, having to enable and switch to using cellular 4G connectivity. Once the user arrives at the destination, potentially their place of work, the device will then associate with another 802.11 WLAN network, changing the connectivity yet again.

In the current Internet, this series of movements and changes to connectivity will disrupt the communication, even though, at any given moment during the communication, the device has one or more than one connectivities available. Despite the connectivities being present, it cannot effectively utilise them to provide consistent connectivity. This is primarily due to three issues:

1. Handoffs across different technologies (vertical handoffs) are not possible since the inter-workings between the different technologies do not exist.

2. Only one interface can be used at a given time — the transport layer state is bound to a single Internet Protocol (IP) address, and since an IP address is bound to a specific network interface, the transport layer can only utilise one interface at a given time.
3. Handoffs across separate networks are not possible without interruption to ongoing communications — changing the IP address will disrupt the transport layer state.

Fundamentally, the challenges lie in the use of IP addresses. The communication ‘session’¹ will break as changes to the IP address will disrupt the transport layer tuple. Applications can either implement a mechanism to automatically reconnect with the new IP address which was acquired or became active as the primary network interface changes, or let the user manually re-initiate the communication as the application level session breaks when the IP address changes. Mobility mechanisms described in Chapters 2–3 can help to continue the communication without the transport layer tuple breaking. However, the performance of handoffs will differ depending on the types of handoffs the mechanisms can perform (soft/hard-handoff). Furthermore, some of them may only work in a certain transport layer protocol as they may be implemented as an add-on to the certain transport layer protocol.

Multihoming-mobility, in such a scenario, for example, may involve utilising cellular connectivity whenever possible or reasonable to improve its throughput or resilience.

In the context of network mobility, devices are often concerned with wireless connectivity, such as IEEE 802.11 variants and cellular technologies like 4G and 5G. With such technologies, the network is provided in cells, a physical area where a specific base-station or an access point provides the service. In most cases, cells overlap; when a mobile device moves between different cells, it often experiences a time duration where two networks are available to it. In the following series of experiments, this duration is referred to as an ‘overlap-period’.

To explore the user mobility scenario with continuous movement, testbeds were developed. Specifically, to further develop mobility and multihoming mechanism using ILNPv6, a development testbed was built, and an evaluation testbed was built to enable empirical evaluation of developed mechanisms.

¹In Transmission Control Protocol (TCP) context, it will be a TCP session, and with User Datagram Protocol (UDP), this will be for connection-based UDP transactions.

4.2 Rationale for testbed-based evaluation and engineering effort

The testbed-based approach was used to conduct evaluation for a number of reasons. Simulation-based approaches offer several benefits, such as flexibility, scale, and speed; simulations can manipulate the evaluation very easily, enable evaluation with larger scale easily, and reduce the time required to conduct evaluations. However, simulations are fundamentally a sandboxed model of the system under test. Therefore, simulations can introduce errors, such as missing effects in the model, limiting the scope to the capability of the sandbox, and most critically, this does not directly evaluate the actual code to be deployed.

One of the focuses of the thesis is evaluating whether mobility-multihoming duality can be realised *without requiring changes to core infrastructures or existing applications*, as stated in the thesis statement in Section 1.5. To evaluate that, the solution is contained within changes to the end-host and the end-host only — it is critical for this evaluation that existing hardware and software function with the mobility-multihoming duality mechanism. Hence, the evaluations conducted in this thesis took the testbed-based approach.

Testbed-based approaches involve substantial engineering effort. The primary engineering effort of simulations is to design and model the behaviour and produce measurements. However, testbed-based approaches, on top of designing the behaviour and conducting measurements, involve a substantial number of tasks; testbeds require hardware management, software compatibility checks, software packaging, handling of changes from upstream developers, and examining and understanding existing software that may require modification. Despite the additional engineering work required, a testbed-based approach was chosen to boost confidence in the solution, examine the crucial part of the thesis statement, and provide opportunities to conduct more evaluations in the future.

4.3 Development testbed

In addition to the evaluation testbed, ILNPv6 was developed using another testbed. The development testbed was primarily used for debugging and testing. The results in Chapters 6–7

were not collected using this testbed; however, the binaries were compiled and packaged using the development testbed as a practical convenience.

The development testbed consisted of two 46-port switches and twelve rack-mounted servers. One of the switches was a gigabit managed switch dedicated for a test network and the other was a 100 Mbps switch for control/management interfaces. The experiment network switch provided non-blocking forwarding and virtual LAN (VLAN) capability, allowing multiple access networks to reside completely separated from one another. Routing was done using six rack-mounted servers with open-source router operating system VyOS, connected in a star topology. The *head-node* handled host management and kernel compilation. The rest of the rack-mounted servers acted as end-hosts with Debian 9.

While this testbed was physically capable of carrying out the evaluation, the experiment testbed in Section 4.4 was used instead. The experiment testbed network consisted of widely used off-the-shelf networking equipment; meanwhile, the development testbed network consisted of rack-mounted servers with open-source router operating systems. To carry out empirical evaluations of ILNPv6 and its continuous mobility and mobility-multihoming duality mechanisms, it is important to use unmodified equipment deployed in *‘real-world’*. Dedicated networking devices often involve optimisation, which has the potential to cause issues with new emerging protocols.

4.4 Evaluation testbed

The following describes the testbed used for empirical evaluations in the following chapters. The evaluation testbed was designed with the goal of providing an environment that is, while isolated, effective in evaluating the functionality of the protocol that can operate over a *standard* commercial networking infrastructure used today. This is a key part of the requirements derived from the thesis statement in Section 1.5 part B1, stating *“without requiring changes to core infrastructures”*. The testbed is designed to evaluate three key aspects mentioned earlier:

‘Handoffs across different technologies’: Handoffs using wired Ethernet will be a proxy to evaluate scenarios where two technologies have no inter-workings established, as they do not have any handoff mechanisms.

‘Only one interface can be used at a given time’: The hosts in this testbed have multiple interfaces — evaluating the mechanism with multiple interfaces is essential.

‘Handoffs across separate networks’: The testbed network consists of four access-networks, one for the Correspondent Node (CN) and three for the Mobile Node (MN). These are separate IPv6 networks connected with static routes.

The user scenario described previously includes a mobile device, which typically does not have powerful hardware. To reflect this, unlike the development testbed, the evaluation testbed consisted of two desktop end-hosts of modest specification with Intel Atom system on a chip (SoC). The testbed network in the development testbed consisted of rack-mounted server units with open source router operating system (OS). On the other hand, the evaluation testbed network was comprised of commercial, off-the-shelf network devices that are deployed in many commercial installations. Those two aspects here were crucial to evaluate the requirements laid out in the thesis statement and address the key issues mentioned above. Specific adjustments for different sets of evaluations are described in respective chapters. In the following sections, the underlying basic configuration of the testbed is described, in terms of its hardware, software, and network configurations.

4.4.1 Hardware and software

The testbed consisted of the following equipment:

Two end-hosts:

Mobile Node (MN): A node that moves across different networks

Correspondent Node (CN): A node that is stationary

Home Agent (HA) for Mobile IPv6 (MIPv6):

A desktop host acting as the HA and the router serving the ‘home-network’ for the MIPv6 host

Four routers:

All commercial products, unmodified, supporting IPv6, but not ILNPv6 aware.

One Ubiquiti Networks Edge Router 6P:

R4 in Figure 4.3, providing access network to CN and providing connectivity to R1–R3

Three Ubiquiti Networks Edge Router X:

R1–R3 in Figure 4.3, providing access networks to MN

See Table 4.1 for further details. Figure 4.2 shows a photo of the testbed. The routers were not modified in any way to accommodate ILNPv6, only functioning as simple IPv6 routers; the routers had Ubiquiti Networks EdgeOS without any additional packages or custom firmware installed. They formed the four access networks and inter-router networks — one for the CN, and three for the MN.

MN and CN hosts were desktop hosts with Intel Atom SoC. Debian 9.7 was installed with two versions of Linux kernel 4.9.52, one with ILNPv6 implementation and one without. The hosts have MIPv6 userland daemon, `mip6d` from `umip.org` installed. To allow name resolution without the domain name system (DNS), a modified `glibc` was used to enable the `/etc/hosts` file to hold ILNPv6 Identifier-Locator Vector (I-LV) entries. The DNS was eliminated from the testbed in order to reduce the complexity of the testbed and to remove an additional variable that may impact the performance, as a DNS lookup is non-deterministic and can impact the behaviour of the end-hosts.

In Chapters 5–6, network benchmark software `iperf2` version 2.0.9 was used to conduct the experiment, generating synthetic payloads. In Chapter 7, `ffmpeg` and `vlc` were used in order to send and receive Real-time Transport Protocol (RTP)/UDP payloads. In all cases, the applications were not modified in any way, and were sourced from the public Debian package repository. They were binaries directly taken from the publicly accessible repository, installed using the APT package management tool as any user would, and they utilised the standard C Socket application programming interfaces (APIs) for all communications. Therefore, the only modifications on the end-host were in the Linux kernel and `glibc`.

Table 4.1 summarises both the software and hardware used in this testbed.

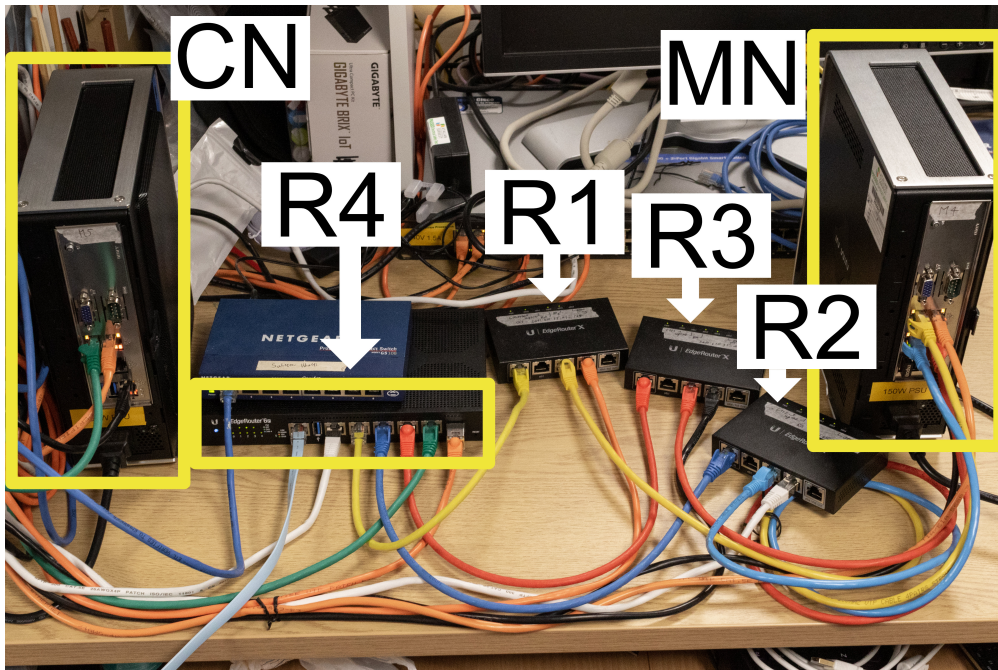


Figure 4.2: A photo of the testbed. From the left side: m5 (CN), R4, R1–R3, and m4 (MN). The switch at the back was purely for control and management of the testbed. Orange and black cables were used for the management network only. The yellow ethernet cables were for network **aa**, the blue ethernet cables were for network **bb**, the red ethernet cables were for network **cc**, and the green ethernet cable was for the CN, network **dd**. The HA for MIPv6 was outside of the frame on the shelf just behind the CN. The physical hardware of HA was the same as the CN and the MN.

	Equipment / software used	Additional information
R1, R2, R3 (R4 (Note 4.))	Ubiquiti Network ER-X	(Note 1.) Software version 1.10.9
R4	Ubiquiti Network ER-6P	(Note 2.) Software version 1.10.7
HA (MIPv6) Used only for MIPv6	Motherboard: ASRock C3558D4I-4L	Intel Atom C3558 (4-core) SoC 4x Gigabit Ethernet 8 GB DDR4 RAM, 223 GB SSD VyOS 1.2.0-rolling+201905212216
	Operating System Mobile IPv6	umip 1.0 http://umip.org (Note 3.)
CN / MN	Motherboard: ASRock C3558D4I-4L	Intel Atom C3558 (4-core) SoC 4x Gigabit Ethernet 8 GB DDR4 RAM, 223 GB SSD
	Operating System Packet Capture	Debian 9.7 with Linux kernel v4.9 ILNPv6 tcpdump 4.9.3 with libpcap 1.8.1
	Mobile IPv6 (Only for MIPv6 Scenarios)	umip 1.0 http://umip.org (Note 3.)

Notes:

1. <https://www.ui.com/edgemax/edgerouter-x/>
2. <https://www.ui.com/edgemax/edgerouter-6p/>
3. Inaccessible at the time of submission
4. In continuous-mobility-iperf experiments, ER-X was used at R4 for the majority of scenarios; one scenario was conducted with ER-6P instead of ER-X but posed no anomalies upon visual inspection of the results.

Table 4.1: Testbed equipment information.

Glibc modification for holding IL-V in the /etc/hosts

The glibc C library was modified to hold I-LV in the /etc/hosts file. Specifically, glibc was modified to parse the custom I-LV syntax as shown below:

```
n-n-n-n.l+1+1+1 hostname
```

The former n-n-n-n is the Node Identifier (NID), that is of the similar format to the unicast IPv6 interface identifier (IID) where “:” was replaced with “-”. The latter l+1+1+1 is the Locator (L64), that is the unicast IPv6 routing prefix where “:” was replaced with “+”. This provides a fixed I-LV used to initialise the ILNPv6 communication. As described, this was built only for the purpose of the experiment. Below is an example of the I-LV entry:

```
d250-99ff-fed1-5a0c.2001+2+dd+dd m5-ilnp
```

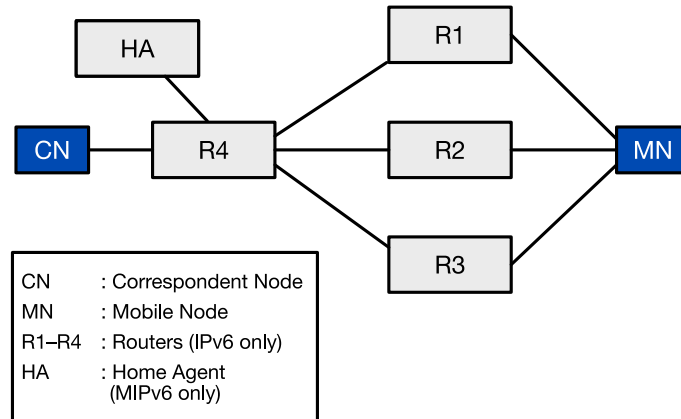


Figure 4.3: A physical topology diagram of the evaluation testbed.

4.4.2 Network

The testbed network was built using the aforementioned off-the-shelf routers and an additional desktop host as the HA for only MIPv6 scenarios. The physical topology of the network is shown in Figure 4.3. The network was a simple IPv6 network with static routes set up between the routers. Each access network was configured as a regular IPv6 network, with stateless address autoconfiguration (SLAAC) [63] and prefix advertised using router advertisement (RA) [64]. In total, four routers provided four access networks and respective routers:

Network aa — R1 — 2001:2:aa:aa::/64

Network bb — R2 — 2001:2:bb:bb::/64

Network cc — R3 — 2001:2:cc:cc::/64

Network dd — R4 — 2001:2:dd:dd::/64

Networks aa–cc were dedicated to the MN, while network dd was dedicated to the CN.

Table 4.2 shows the RA timing parameters used in the access networks. The intervals were set as short as possible, and the lifetimes were configured three times the interval duration. Specifically, the valid-lifetime was set at three times the maximum interval, while the preferred-lifetime was set at three times the minimum interval. The routes were configured statically within the respective routers; more specifically, R1–R3 routed ::/0 to R4, and R4 held the appropriate

Parameter	Value (sec.)
max-interval	5
min-interval	3
valid-lifetime	15
preferred-lifetime	9

Table 4.2: Lifetime and interval settings of router advertisement on access networks.

static routes for respective access network’s prefix on R1–R3. Routers and end-hosts were connected using gigabit Ethernet instead of 802.11 WLAN. This eliminated the environmental radio interference potentially impacting the result, and the potential fluctuation of throughput becoming the bottleneck, isolating the network protocol performance from underlying system effects. The IP address/I-LV lookup was executed by hosts using an extended `/etc/hosts` file and the modified `glibc` to simplify the testbed, eliminating a potential source of latency at the start of the communication session and eliminating the need for maintaining a DNS server in the testbed.

The HA was connected to the R4, acting as the router for the MIPv6 ‘home network’. It was a desktop host with Intel Atom SoC — the same type of machine as the CN and the MN — with VyOS, an open-source Vyatta-fork router operating system, with the userland MIPv6 daemon, `mip6d` installed. The userland MIPv6 daemon was acquired from <http://www.umip.org>, which was no longer accessible in September 2021 and remains inaccessible at the time of writing.

4.5 Evaluation scenarios

To carry out an empirical evaluation of the mechanism, two evaluation scenarios were designed based on the user scenario described in Section 4.1, where a user moves through different networks employing different technologies while a communication session is ongoing. The two evaluation scenarios were a continuous mobility scenario and a mobility-multihoming duality scenario. The continuous mobility scenario was designed to compare ILNPv6 and MIPv6 in Chapters 5 and 7, evaluating how the two mobility mechanisms behave in a scenario where the MN performs handoffs one after the other during a single communication session. The mobility-multihoming duality scenario was used in Chapters 6 and 7 to conduct empirical evaluations of the mobility-multihoming duality mechanism where individual network cells from the user scenario have much

larger areas, allowing more than two cells to be available to the MN during part of a single communication session. The scenarios shown below were designed to utilise the topology shown in the Figure 4.3 in Section 4.4.2 emulating the movement of mobile hosts.

4.5.1 Continuous mobility scenario

The continuous mobility scenario considered the situation similar to the user scenario in Section 4.1, where a user moves during a communication session through different wireless network cells, primarily using a single connectivity at a time. To evaluate such a scenario, the setup utilises the three access networks available to the MN to ‘move’ across them continuously one after the other in a cyclic fashion.

1. Enable one interface on both the stationary CN and the MN
2. Begin communication between the two hosts
3. Trigger a handoff:
 - (a) Enable an additional interface
 - (b) Wait for a set handoff-duration
 - (c) Disable the previous interface
4. Repeat step 3 until the experiment run ends

Figure 4.4 shows how the MN moves in such a scenario.

4.5.2 Mobility-multihoming duality scenario

In order to evaluate the mobility-multihoming duality mechanism, the following scenario was developed.

Similar to Section 4.5.1, the MN has access to three networks and moves through various wireless cells. Compared to the continuous mobility scenario, the *cells* have ‘grown’ in this scenario, which create a larger areas of overlap, allowing the MN to utilise three cells simultaneously. In this scenario, the MN moves one way and then it moves the other way within the 120 seconds communication session. See Figure 4.5 for an illustration of the movement.

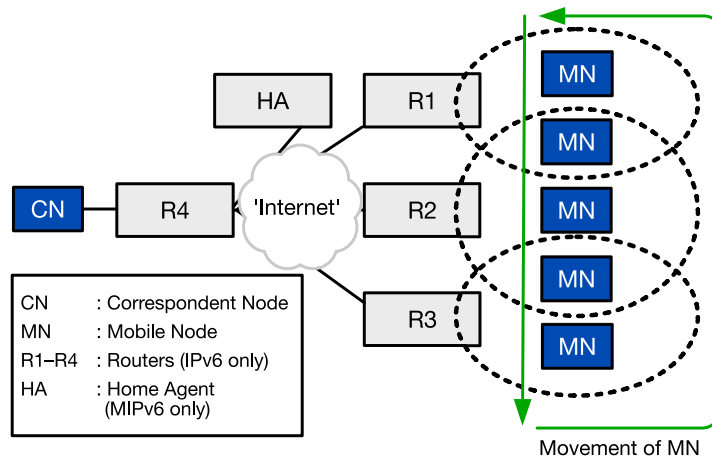


Figure 4.4: A scenario diagram for continuous mobility evaluation. The movement of the MN is shown with the green arrow. The host will move across networks served by R1–R3, and back to R1 again continuously until the communication ends.

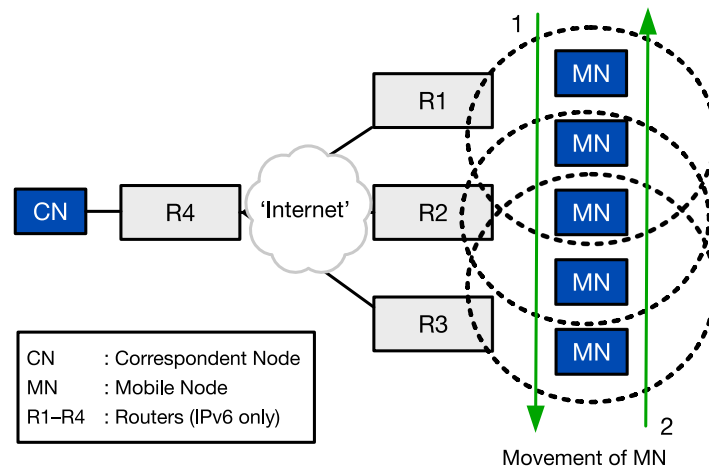


Figure 4.5: A scenario diagram describing host movement for the mobility-multihoming duality scenario. The arrow labelled 1 is the first movement the MN carries out: moving from network aa on R1 to network cc on R3. The arrow labelled 2 shows the second set of movement where the MN returns from network cc back to network aa.

The scenario will be described in further detail in Chapter 6, with regards to how the use of interfaces was controlled and how host ‘moves’ through the network, that is, how the movement of the MN was emulated.

4.5.3 Executing evaluation scenarios

To execute the evaluation scenarios, the following general method was used.

The evaluation consisted of the following steps:

1. Configure the testbed according to the scenario
2. Execute scripts to begin data transfer, collection, and evaluation of scenario movements
3. Collect the captured data from each end-host
4. Analyse the collected data

In the context of the evaluations, a run is defined as ‘one communication session’, which occurs at step 2 in the list above. The following section describes how the movement is emulated during each run.

4.5.4 Emulating movement

As described, the ethernet was chosen as the medium used throughout the testbed network, including the access network. While wireless connectivities are generally more likely to be used with continuous mobility scenarios, wired connectivities were used to reduce potential interference from the surroundings during the data collection as it was a comparative study of handoff performance at the network protocol. To emulate the ‘movement’, the aforementioned scripts were used to enable and disable interfaces. Enabling the interface using the `ip` command triggers a layer 2 link-up on a gigabit ethernet interface, analogous to establishing wireless radio connectivity, which will be followed by an RA, triggering the host to configure the interface’s layer 3 connectivity accordingly. Disabling the interface in the OS is analogous to leaving the cell that the particular interface was connected to in the wireless context. More specifically, link-up is initiated by the `ip link up` command and link-down is initiated by the `ip link down` command.

The scenarios were conducted using bash scripts to automate the actions taken on the end-hosts for repeatability. Primarily, the scripts performed the following tasks:

1. Prepare data collection directory
2. Begin data collection programs
3. Begin data transfer
4. Enable and disable interfaces using `ip` commands according to the movements to be emulated
5. Terminate data collection
6. Reset interfaces

The bash script was copied to the end-hosts and executed remotely.

4.5.5 Overlap duration

The duration of handoff was set to **ten seconds** in all of the evaluation shown in the following chapters. Since this thesis involves comparative studies between ILNPv6 and MIPv6, it was important to eliminate failures and provide a condition that enabled a side-by-side comparison of the two mechanisms reliably. From inspecting the behaviour of the testbed and the relevant mobility mechanism, the ten second ‘overlap’ duration was the minimum duration needed to provide enough time for a handoff to complete considering various factors.

The following is a list of factors impacting the time required for a host to complete a handoff while connected to two networks:

1. Layer 2 link-up
2. Layer 3 IPv6 connection setup
 - (a) RA
 - (b) stateless address autoconfiguration (SLAAC)
 - (c) duplicate address detection (DAD)
3. Mobility mechanism signalling

The duration required for item 1 is unique to a combination of factors including the specific hardware, the driver, the layer 2 protocol, and the surrounding environment for some cases.

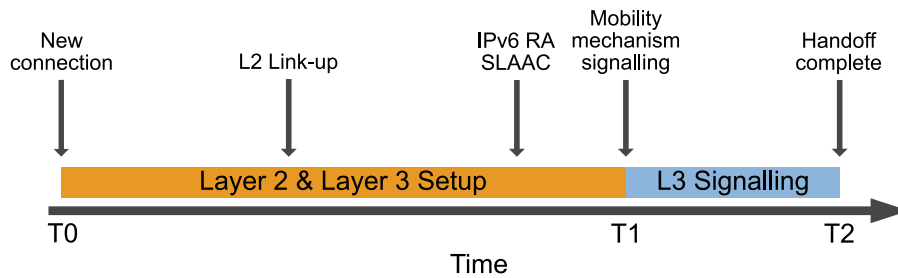


Figure 4.6: A diagram illustrating an overlap duration and events during the overlap duration.

Item 2 involves interface configuration, DAD, and neighbour discovery. Since the scope of the evaluation is for a mobile end-host, we expect the router to be discovered via the RA and the interface to be configured accordingly using SLAAC, instead of static interface configuration. The time it takes for 2 to complete can be controlled using RA configuration; however, it still has some underterministic behaviour due to how the RA is required to be sent at a random duration between the maximum and minimum interval as defined in [64].

From the perspective of network layer protocols such as ILNPv6 and MIPv6, the duration waiting for 1 and 2 cannot be distinguished; i.e. it is not possible for the protocols to distinguish how long it has taken for each phases of the connection setup. On the other hand, once the layer 3 connectivity is established, the latency of handoff is dependent on the mobility mechanism and its signalling latency. Therefore, in order to determine the required overlap duration to reliably complete handoffs for both MIPv6 and ILNPv6, two durations were considered: (A) the time it takes for layer 3 connectivities to be available after initiating link-up on the interface, and (B) the time it takes for the mobility mechanism to complete the handoff. Figure 4.6 illustrates the events during overlap duration. In this context, T0–T1 is determined by the system effects, such as lower layer connectivity establishment, router configuration, and host configuration; and T1–T2 depends on the mobility mechanism.

To determine the time required for layer 3 communication to become available to the end-host, a small evaluation was conducted. The script determined the time that the interface was offline by sending Internet Control Message Protocol v6 (ICMPv6) echo packets from an interface, and by disabling and re-enabling the interface immediately one after the other. An `ping6` program was used to send the ICMPv6 echo packets every 10 ms. The program output

provided timestamp for every reply it received, which was analysed to identify gaps in reply to ICMPv6 echo packets.

After measuring 100 interface on/off, the maximum gap duration was 6.6s and the minimum was 4.5s.² This identified the maximum duration for component (A) of the handoff latency.

The existing work by Ditchaphong Phoomikiattisak and Saleem Bhatti determined the signalling latency for both MIPv6 and ILNPv6 [65]. In Section 6.1.4, Figure 16 in [65] shows handoff delay without the lower layer impacts such as items 1 and 2 mentioned earlier in this section. Figure 16 in [65] shows several scenarios, such as handoffs across local area network (LAN) to LAN, LAN to wide area network (WAN), and so forth. In all cases, the longest delay was due to MIPv6 with Route Optimisation (RO), with the maximum reaching near 3s. The work has also shown that MIPv6 performance deteriorated very quickly as soon as its signalling is lost.

Since component (A) (T0–T1 in Figure 4.6) requires up to 6.6s., and component (B) (T1–T2 in Figure 4.6) requires up to 2.8s, in total, 9.4s the overlap duration to ensure a stable handoff for both MIPv6 and ILNPv6 was determined to be **ten seconds** for this testbed. This is to ensure that MIPv6 completes its handoff, as ILNPv6 requires ~ 1 round trip time (RTT) for T1–T2 duration.

4.5.6 Path round trip time (RTT) configuration

As shown in Figure 4.3, the MN has access to three paths to reach the CN. In the subsequent chapters, the evaluations do not involve variable RTTs across the three paths. Instead, the RTT between the two hosts remain homogenous on all paths. This was chosen as the evaluations conducted in this study are not ‘absolute’ performance analyses. In addition, a specific combination of RTT presents a very specific scenario, especially for TCP. The comparison of a handoff across the two distinct paths with different RTTs was shown in [65], and [66]. The work titled “Control Plane Handoff Analysis for IP Mobility” [67] analytically shows what happens when loss/RTT variation exists across the two paths the handoff is performed.

An additional two RTT configurations are selected to emulate the communication between the hosts at different topological and geographical distances. No added RTT represents the scenario where two hosts reside in the same LAN. An additional 20ms is used to emulate the

²See Appendix B, Section B.2 for a boxplot showing the distribution.

“national” RTT, and an additional 200ms is used to emulate the “inter-continental” RTT. A `ping6` program was used to measure the RTT, in order to determine the required additional RTT to emulate the topological and geographical distances the hosts may have. Section B.3 in Appendix B shows the RTT measurements from a domestic IPv6 network to hosts at various locations to determine the appropriate RTT representing the relevant scenarios.

4.5.7 Payloads

Since the evaluation involved network communication, payloads were needed in order to send and receive data over the network. For experiments in Chapters 5–6, to observe packet level network protocol performance, the network benchmark tool `iPerf2` was used to produce a synthetic payload. `iPerf2` was configured to generate 10 Mbps of traffic, which was sent and received at both sides. `iPerf2` was used for both TCP and UDP data transfer. The UDP `iPerf2` contained a sequence number inside each UDP packet payload, which was used to analyse loss and misordering. In order to prevent packet fragmentation, the command-line options of `iPerf2` was adjusted accordingly, as the default options for `iPerf2` especially caused issues when using MIPv6. In terms of data payload, the default synthetic payload generated by `iperf` was used in Chapters 5–6, providing consistent fixed throughput. Since Chapters 5–6 focuses on packet level Quality of Service (QoS), the use of synthetic payloads was appropriate.

Contrary to Chapters 5 and 6, Chapter 7 involves Quality of Experience (QoE) assessment at the application-level, which requires a more *realistic* payload. Therefore, in Chapter 7, two video clips sent over RTP/UDP were used. The clips were packaged in a commonly used MPEG Transport Stream (MPEG-TS) container with H.264 codec, encoded in variable bit rate (VBR), emulating a real-time video application payload — see Section 7.4.1 for further details. By using a real-time video, which is a ‘*real*’ payload used in *real-life*, with existing common applications, the effectiveness and performance impact of ILNPv6 to existing applications was evaluated using objective QoE metrics.

4.6 Data collection and metrics

To evaluate the performance and the capability of ILNPv6, the following metrics were used in experiments across the three sets of evaluations:

- Throughput
- Packet loss
- Packet misordering

To measure packet-level metrics, in all of the evaluations, packets were captured using `tcpdump`, saved as a `pcap` file, both at the sender and the receiver. To reduce the size of the `pcap` files, packets were captured with the snap length (`-s`) option to eliminate the latter part of the transport layer payload. For UDP and RTP/UDP packets, the beginning part of the UDP payload contains the sequence number of the packet, which was preserved by setting the snap length accordingly upon initial observations.

Processing pcap files To process the `pcap` files, they were first converted into comma-separated values (CSV) files using `tshark`³. R scripts were used to process the converted CSV files to carry out the following steps:

- Filter traffic
- Calculate throughput
- Parse and analyse sequence number
- Visualise throughput and sequence number progression
- Output throughput and sequence number stats of the run

³command line utility portion of network packet analyser/capture tool Wireshark — version 2.6.20 on Debian 9

Throughput A throughput measurement for a run was taken using the sum of the individual IPv6 packet payload size of the run divided by the duration of the run.

$$\text{Throughput} = \frac{\sum p}{t_{\text{end of flow}} - t_{\text{beginning of flow}}}$$

where:

p = IPv6 payload length of a packet in the flow

t = time in seconds

The sum of IPv6 payload length was calculated from packet length field in IPv6 packet header. The duration of the run was measured by the first and the last packet of the flow, which was determined by packets filtered by ports, transport layer protocol, and source/destination address/I-LV to correctly identify the flow that was received at the host rather than sent from the host. The throughput measurement was done using what was received at the receiver, to observe successfully received throughput, as seen at the receiver. In Chapters 5–6, iPerf2 was used to generate a synthetic payload; although iPerf2 does have throughput measurement capability, it was not used. Calculating the throughput from the packet captures enabled easier comparison between different payloads and prevented potential error; iPerf2 is a diagnostic/benchmarking tool for IPv4 and IPv6 without any awareness of ILNPv6 L64 or NID. Since ILNPv6 makes use of names differently, there was a chance that the measurement may not be reported correctly using iPerf2's analysis mechanism. In addition, developing a way to calculate throughput across different transport protocols by using the payload length in IPv6 header enabled the throughput measurement in Chapter 7 with RTP/UDP traffic.

Sequence number analysis Similar to the throughput, sequence number observation and analysis were done based on what the receiver had received successfully, recorded in the packet capture. With UDP-based traffic, misordering were identified using sequence numbers in the payload part of the captured UDP packets. iPerf2 includes a sequence number in every packet, which iPerf2 itself uses to identify packet delivery stats. Specifically, the sequence number is in the first four bytes of the iPerf2 UDP payload.

TCP traffic required analysis using the sequence numbers observed at both the sender and the

receiver. The sequence number is part of the TCP packet header, however they start at a random value and does not increment consecutively as it advances based on the bytes sent. A loss will trigger a resend; however, resends may be caused by two different events depending on the type of loss. If the packet carrying the payload is lost, the sender will observe a duplicate sequence number and the receiver will only observe one. Meanwhile, if the acknowledgement packet to that sequence number is lost, the sequence number will be repeated at both sides, while the acknowledgement number will be observed twice at the receiver of the payload packet, and the original sender of the payload will only observe the acknowledgement packet once. In addition, a duplicate sequence number of the packets with a payload may be observed at both sides if there is a misordering of the packets. Here, the sender-only-duplicates are used to determine the loss of packets with an actual payload.

With UDP-based traffic, loss was derived from the difference between the total sent packets and the total received packet count. Since UDP-based traffic does not have resends, observing the packet counts at the sender and the receiver was adequate to detect loss.

Misordering analysis is similar with both UDP and TCP. Misordering was detected by observing when the difference between the sequence number of a packet has decremented, rather than incremented, and if it has decremented, the sequence number has not wrapped.

Both the loss and the misordering are presented as ratio. Specifically, the misordering rate is a ratio of misordering count against the total packets received carrying the transport layer payload, while the loss rate is a ratio between the lost packet count to the total sent packets. They were calculated as follows:

$$\text{loss rate} = \frac{n_{\text{loss count}}}{n_{\text{total sent packets}}} \quad (4.1)$$

$$\text{misordering rate} = \frac{n_{\text{misorder count}}}{n_{\text{total received}}} \quad (4.2)$$

4.7 Software engineering summary

In order to utilise the physical testbed and conduct evaluation outlined in the previous sections, a in-kernel implementation of ILNPv6 is required.

While the ILNPv6 implementation with mobility feature for Linux kernel exists from the previous PhD study, the implementation required to be migrated to a more modern version of the kernel before further evaluation or development can be conducted. In 2017, the latest long-term support (LTS) Linux kernel version was at version 4.9. Therefore, the ILNPv6 implementation in Linux kernel 3.9 has been migrated to version 4.9 with required adjustments. The changes to ILNPv6 implementation required to enable continuous mobility occurred during the migration. The changes required for mobility-multihoming duality was done after the migration and evaluation of continuous mobility mechanism.

4.7.1 Existing files modified in the Linux kernel

While there are only two new files, as mentioned in Section 3.2, existing IPv6 mechanisms were modified to accommodate Identifier Locator Network Protocol (ILNP), thus following files have been modified:

- `include/linux/icmpv6.h`
- `include/linux/skbuff.h`
- `include/net/ip6_checksum.h`
- `include/net/sock.h`
- `include/uapi/linux/in6.h`
- `include/uapi/linux/netlink.h`
- `net/ipv4/tcp.c`
- `net/ipv4/tcp_output.c`
- `net/ipv6/Makefile`
- `net/ipv6/addrconf.c`
- `net/ipv6/af_inet6.c`
- `net/ipv6/datagram.c`
- `net/ipv6/exthdrs.c`
- `net/ipv6/icmp.c`
- `net/ipv6/inet6_hashtables.c`
- `net/ipv6/ip6_checksum.c`

- `net/ipv6/ip6_input.c`
- `net/ipv6/ip6_output.c`
- `net/ipv6/ndisc.c`
- `net/ipv6/raw.c`
- `net/ipv6/route.c`
- `net/ipv6/tcp_ipv6.c`
- `net/ipv6/udp.c`

In total, twenty-eight functions were modified in those files to migrate. While the few only required a simple migration of code, most required restructuring of the ILNPv6 related code and some did not exist back in kernel version 3.9.

The following steps are taken for each functions in order to migrate the implementation.

1. Identify the equivalent part of the source code in version 3.9 and 4.9 where ILNPv6 specific changes are present
2. Identify if any upstream changes are present
3. If any upstream changes are present, identify the changes and its impact
4. If changes are significant, identify the reasons by investigating through commit messages and relevant mailing list thread
5. If required, design new logic to accomodate changes — in some cases, requiring finding where a specific section of code have been migrated to
6. Implement/migrate ILNPv6 code, compile and verify the changes

Primary changes made to transport protocol related functions are introducing locator handling for ILNPv6 packets. Specifically the code enabled masking of locators before the I-LV is processed, and restore them accordingly. Socket lookup must be done with only the NID, and checksum calculation must be done by only using the NID if and only if this is for ILNPv6 communication.

After the migration to kernel version 4.9, the implementation was extended to support continuous mobility, as shown in Chapter 5. The implementation used in Chapter 5 is published

as `ilnp-public-1` [68], the first ever public release of ILNPv6 prototype implementation on Linux kernel implementing the continuous mobility mechanism.

Following Chapters outlines respective design changes and engineering efforts to enable respective additions to ILNPv6.

4.8 Summary

The testbeds presented here were used to develop and evaluate the effectiveness of ILNPv6 in enabling true mobility and multihoming duality. By using both synthetic and ‘real-life’ payloads, ILNPv6 and its extended functionalities will be evaluated using off-the-shelf networking equipment with widely-used open-source applications. In order to enable the evaluation of ILNPv6, in-kernel implementation of ILNPv6 was migrated from Linux 3.9 to Linux 4.9. The following chapter presents how a smooth continuous mobility can be enabled using ILNPv6; this an important step closer to enabling true mobility-multihoming duality.

Chapter 5

Continuous Mobility

Previous work on Identifier Locator Network Protocol v6 (ILNPv6) has shown a smooth handoff between two networks and two interfaces happening just once in a single communication session [58]; the work compared hard and soft handoff mechanisms using ILNPv6 against Mobile IPv6 (MIPv6). However, the mobile devices often move beyond two networks, perhaps continuously, and potentially returning to where they came from within a single communication session. While the previous work demonstrated that ILNPv6 is capable of enabling a single handoff, it did not demonstrate any scenarios where multiple handoffs occurred one after the other continuously. This chapter shows comparisons against MIPv6 in a continuous mobility scenario, a scenario which modern mobile devices are likely to encounter.

5.1 Changes from previous work on ILNPv6

The work described in this chapter builds onto the work of Ditchaphong Phoomikiattisak [58]. This section clarifies the changes and contributions compared to the previous work.

5.1.1 Scientific contributions

The previous work of Ditchaphong Phoomikiattisak [65, 58] showed a single handoff, in a single direction, between two specific networks during a single communication session. Fundamentally, [65] provided detailed analyses of the single handoff performance at the network protocol level.

Overall, this chapter presents two scientific contributions. One of the contributions presented in the following sections shows the improved mobility mechanism — enabling a truly mobile node that can handle mobility across multiple networks seamlessly and continuously. Another contribution here is the extended evaluation. The previous work focused only on a single handoff, however this work presents a sequence of handoffs during a single communication session to show a truly mobile node that can move over different networks beyond a single handoff. This provides a wider view of the flow, demonstrating how multiple handoffs impact the whole communication session.

5.1.2 Changes from previous ILNPv6 implementation in Linux kernel

The previous implementation of ILNPv6 on Linux kernel [58] was implemented in Linux kernel version 3.9. The previous implementation mainly showed the following:

1. Hard and soft-handoff mechanisms with a single locator in the Locator Update (LU) message (i.e. movement across a single network boundary)
2. A single handoff in a communication session

In this work, the ILNPv6 was implemented based on [58] on Linux kernel version 4.9, which became the long-term support (LTS) kernel in 2017. The extended ILNPv6 implementation used in this chapter enabled the following:

1. Continuous communication across multiple handoffs within a single communication session
2. Continuous movement across multiple network boundaries

To realise a true mobility solution, it is crucial to enable mobility beyond a single handoff, and to make sure it functions with more networks than just two. Between Linux kernel version 3.9 and 4.9, there has been major changes to the Internet Protocol (IP)v6 networking implementation in the Linux kernel itself. This required some adjustments and adaptation, specifically, with regards to changes in the socket lookup process and functions that had been merged. However, overall, most of the changes from the previous version were done to enable continuous mobility. The ILNPv6 implementation presented in this chapter is published as `ilnp-public-1` [68], the first public release of the prototype ILNPv6 implementation.

5.1.3 Multiple handoffs

The implementation was extended to enable more than one handoff; this was done by updating locator handling, updating the network configuration, and configuring sysctl variables in the end-host. In addition, the network has been re-configured to enable correct handling of Locators (L64s) to allow the Mobile Node (MN) to return to a previously associated network.

The previous system implementation of ILNPv6 did not allow a mobility beyond a single handoff. The primary issue was the L64 expiry timing, which is dictated by the prefix lifetime in the router advertisement (RA). The L64 status in the ILNP Communication Cache (ILCC) becomes VALID when the interface receives an RA, and uses the unicast routing prefix as the L64. When a packet is sent and received using a VALID L64, the L64 becomes ACTIVE. If another L64 is detected on a different interface, while an ACTIVE L64 is present, this is when a soft handoff is triggered. First, the new L64 becomes VALID. Second, an LU is sent to the Correspondent Node (CN), with the new L64. Third, once the LU is sent, set the new L64 as ACTIVE, and set the previous L64 as VALID. The previous L64 state will not change with RAs while it is VALID, as that will trigger another handoff back to the L64, causing a ping-pong effect. Therefore, before triggering another handoff, the L64 must expire.

As discussed in [58] and [65], timely discovery of the RA is important for mobile hosts; however, equally, for the most up-to-date state of connectivity at the end-host, the lifetime of the prefix is equally important. As a mobility system, the appropriate access network configuration is essential. Here, for the implementation of ILNPv6, not only the interval is crucial for continuous mobility but also the corresponding lifetime duration. Therefore, the interval and the lifetimes were specifically set to the values as presented in Table 4.2.

5.1.4 Locator Update and L64 state

The LU sending mechanism has been altered to ensure the timings of L64 state changes match the LU signalling. The previous implementation changed the state of L64s as soon as an RA had been received on the “new” interface. This led to a small amount of packets with the new L64 to be sent out before an LU — which signals the L64 change — was sent. This means This

implementation updated the state management logic to prevent the L64 state change until the LU has been sent out.

As described in Section 5.1.3, the L64 has to first become VALID, then ACTIVE; and for an ongoing communication, the LU has to be sent before actively transmitting from the new L64. This is because the CN must know about the new L64 before receiving packets from the new L64.

The L64 state management has been updated to ensure that the state of the new L64 becomes VALID before the LU is sent, then becomes ACTIVE after the LU is sent. This does not completely eliminate the chance of loss, as the path that the LU is sent from could have a much larger round trip time (RTT) than the new path that the new L64 is associated with. However, on the other hand, waiting for the Locator Update Acknowledgement (LU-Ack) is also problematic, as with longer RTT path, this will delay the handoff completion, potentially leading to the previous interface being disconnected before the data flow migrates to the new interface, causing disruptions to flows from the MN. Therefore the new L64 should not wait for the LU-Ack and should be put to use as soon as LU is sent.

This implementation of ILNPv6 has some modifications to the L64 state transitions compared to RFC6741 [54]. Figure 5.1 shows the states implemented and how they transition with this particular version. This is based on the states presented in Section 3.4.3 in Figure 3.4. AGED is one of the L64 states defined to accommodate situations where the local L64 has expired due to the lack of RAs while communication is in progress as shown in Figure 3.4 on page 46. In a situation where an RA is delivered intermittently, the AGED state could allow data flow to continue and return to the ACTIVE state once the RA resumes. However, this mechanism was omitted to simplify the implementation.

Scenarios that could lead to the AGED state are uncommon failure states. In addition, since the RA mechanism is mandated in the router, it is extremely rare for a router to stop sending RAs while the rest of the router is functioning. If the RA mechanism is disrupted, it is much more likely for the packet forwarding and the network to be disrupted. In the context of this evaluation, such situation did not occur; therefore, the results were not impacted by this behaviour.

Without the AGED, the L64 will transition as follows:

1. VALID is set as the initial state when the L64 is received via an RA
2. Data flow begins on the L64:
 - (a) The L64 becomes ACTIVE
 - (b) Once data flow ends, the L64 returns to VALID
3. The RA is lost, the L64 lifetime runs out:
 - (a) The L64 becomes EXPIRED
4. The RA is received after 3a; L64 returns to VALID

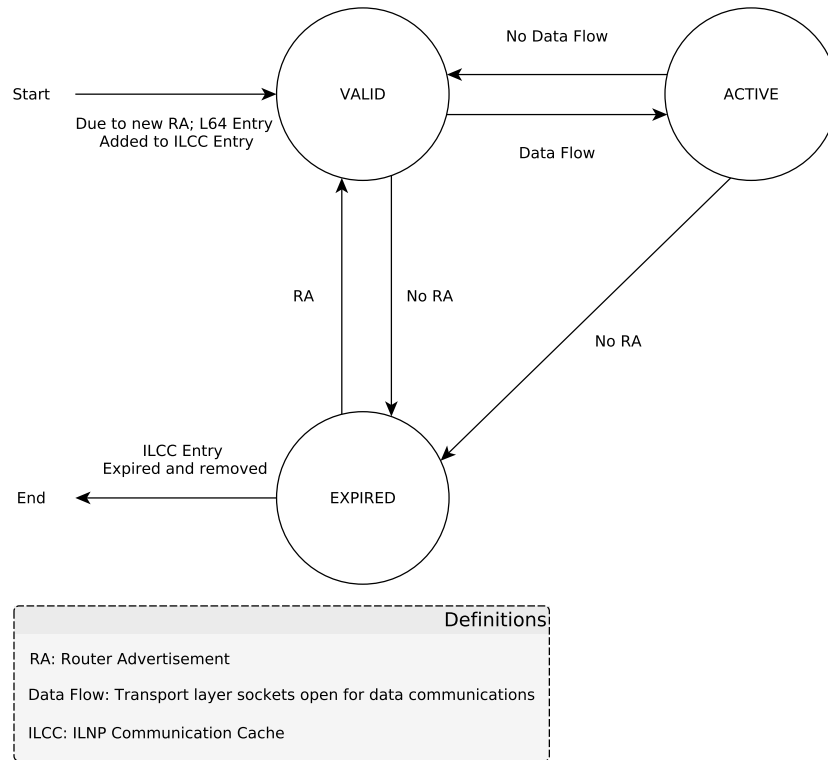
5.1.5 Socket lookup

One of the changes was made on the socket lookup mechanism. One of the files `inet6_hashtables.c`, holds the function for the IPv6 Transmission Control Protocol (TCP) socket lookup mechanism. In order to enable Node Identifier (NID)-only lookup, when ILNPv6 is enabled via `sysctl` option, The 64-bit L64 portion of the 128-bit IPv6 address is masked before the lookup and recovered before resuming the packet processing. At the end of lookup, the masked L64 must be recovered before leaving the function. However, the previous implementation did not recover it in some of the cases, leaving the socket without the L64 portion of the address, which is needed for communication with IPv6 only hosts. This has been resolved with the newer implementation, along with adapting to upstream changes.

The previous implementation relied on a separate cache data structure to look for IPv6 addresses that are actually Identifier-Locator Vectors (I-LVs). This implementation added `in_ilcc()` to cover corner cases that may occur at the beginning of the communication session to look up the full ILCC to correctly identify ILNPv6 I-LVs.

5.1.6 Interface selection for NID generation

One of the assumptions which the previous implementation made was the interface name to generate the NID from. Specifically, in `addrconf.c`, the previous implementation generated the NID from the `eth0` interface. This implementation has removed the hard-reference to `eth0`.



VALID: Locator within its lifetime, with no active data flow

ACTIVE: Locator within its lifetime, with active data flow

EXPIRED: Locator beyond its lifetime, with no active data flow

‘No Data Flow’: Transport layer session(s) using the locator has ended

‘Data Flow’: Transport layer session(s) using the locator has begun; the locator is in active use

‘RA’: Receiving router advertisement with a prefix used for the locator

‘No RA’: The router advertisement with a prefix used for the locator is not received beyond the prefix’s lifetime

Figure 5.1: An L64 state diagram showing how the L64 state transitions without the AGED state.

Instead, the updated `addrconf.c` looks for the first interface that is not a loop-back interface, and uses its interface identifier (IID) as the NID, by setting them in the respective IID fields on the other interfaces. This enabled portability of the kernel to hosts with different interface naming schemes, such as the systemd “predictable network interface name”¹ naming scheme.

5.2 Evaluation Setup

In order to evaluate how ILNPv6 performs under the continuous mobility scenario, the following comparative evaluation against the standard mobility solution was designed. A physical testbed was used to conduct the evaluation comparing two IP mobility mechanisms: Mobile IPv6 (MIPv6), the Internet Engineering Task Force (IETF) standard and the general IP mobility solution; and Identifier Locator Network Protocol (ILNP), specifically Identifier Locator Network Protocol v6 (ILNPv6).

As described in Chapter 4 Section 4.4.1, the testbed consisted of the following equipment: two end-hosts, a Mobile Node (MN), and a Correspondent Node (CN); and four off-the-shelf routers, as well as four Ubiquiti Networks Edge Router X². For MIPv6 scenarios, an additional host acting as a Home Agent (HA) was used to track the MN’s Care-of Address (CoA) and manage its mobility signalling.

5.3 Procedure

The testbed was setup according to Chapter 4 Section 4.4. Configurations of hosts and network devices were inspected before starting a set of runs. Especially for MIPv6 scenarios, the additional host (HA) and the userland daemons (`mip6d`) were inspected accordingly, specifically, whether they were running with the appropriate configuration file or not. After the initial check, a set of runs were initiated with a system restart between each run on the end-hosts.

¹https://systemd.io/PREDICTABLE_INTERFACE_NAMES/

²One of the scenario was executed using Edge Router 6P at R4 in Figure 4.3, however no anomalies or difference was observed upon conducting visual inspections of the data collected.

5.3.1 Emulating Movement

The emulation of movement was carried out as described in Chapter 4 Sections 4.5.3–4.5.4, using bash scripts enabling and disabling the interface. The continuous mobility scenario used in this evaluation is described in Chapter 4, Section 4.5.1.

While the definition and triggers of handoffs are different between the two protocols, to conduct a comparative study, the timings of the link-up command and link-down command were kept identical. Specifically, the intervals for triggering the link-up of the ‘new’ interface and the link-down of the ‘old’ interface were set to ten seconds as both protocols’ handoffs were reliably completed within ten seconds during the initial observations; specifically, MIPv6 required the duration to be longer to reliably complete its handoffs.

Mainly, the following three factors are involved in the duration between the beginning of the handoff and the completion of the handoff. First of all, there is a small delay after the link-up has been triggered using the `ip` command in the experiment script till the layer 2 link is established, as layer 2 link-up phase takes some amount of time. Another factor is the router advertisement (RA), more specifically, the time between link-up and the router solicitation (RS) being sent, and the time between the router solicitation (RS) and arrival of the first RA. After acquiring the prefix, duplicate address detection (DAD) [62] can impact the timing between the address configuration and the beginning of the actual communication, which is a non-deterministic duration up to several seconds. Section 5.5.3 discusses the handoff latency in further detail. In this experiment, optimistic DAD [69] was enabled at the end-hosts as they are mobile hosts expecting frequent movements. Naturally, the network protocol signalling greatly affects the handoff duration and performance, such as the number of signalling required, as well as the latency of the paths it must take to reach the other side. With all of the above factors considered, the ten second interval for interface up-down duration was chosen for this evaluation. The emulated movement of the nodes were configured to be identical between the two mobility solutions, allowing side-by-side comparison of the whole flow.

5.3.2 Metrics

To evaluate ILNPv6 and MIPv6 in the continuous mobility scenario, in addition to the metrics described in Section 4.6, additional metrics were used.

In a continuous mobility scenario, one of the factors that could impact the ‘quality’ of data communication is the changes in throughput during the handoff. While many of the network transactions are asynchronous and not necessarily time-sensitive, more applications on the Internet are time-sensitive, and synchronous in the usage pattern. An example is real-time-audio/video applications. In addition, even with a less time-sensitive application, with enough disruption to the throughput, unstable throughput may affect its performance and the user’s experience. For example, a Video on Demand service streams videos in chunks and buffered at the host, and thus, it is not strictly time-sensitive; however, if the delivery of chunks delays beyond what is buffered at the end-host, the playback may deplete the buffer, causing the player to pause while it waits for the video chunks arrive again, greatly impacting the user’s experience of the video playback. Disruptions and delays during the handoff can also affect non-time-sensitive applications, especially in prolonged transactions where the device is more likely to encounter handoff, such as large file transfers — if the disruptions last too long, they can lead to session timeouts, such as TCP socket timeouts, breaking the entire session, defeating the purpose of the handoff mechanisms entirely. Therefore, the ‘smoothness’ of the handoff and avoiding gaps in data transfer, that is, maintaining consistency in throughput and reducing disruptions to throughput, are both important for continuously moving device and the user, regardless of the time-sensitiveness of the application.

In this set of evaluation, the focus remains the packet level Quality of Service (QoS) metrics. To further investigate the application level impact, Chapter 7 evaluates the impact of handoff performance on the objective Quality of Experience (QoE) metrics using real-time video applications.

Throughput distribution One way to observe the smoothness of throughput is by inspecting the distribution of throughput. While it is expected for the throughput to be at zero at the start and the end of the communication session, and maximum to reach potentially beyond the target throughput (for example, buffered packet bursts and potential sample boundary effect in interval

aggregation), the first, second, and third quartile are expected to be similar. In other words, those values should be similar if the handoff did not disrupt the throughput, impacting the distribution of received throughput. To measure the distribution of throughput, inter-quartile range was taken, observing the range of values between the 75th percentile and the 25th percentile.

Throughput gap Another metric to observe the handoff smoothness is the zero-throughput duration; when the delivery of the packet experience a gap. This also heavily impacts the resulting distribution of the throughput. When the data transfer is disrupted completely, where no packets are being delivered, a range of events may occur; application requiring those payload may delay in its response, change in QoE, or potentially a timeout of the session. Therefore, a cumulative duration of disruption and distribution of the observed zero-throughput duration was used to observe the gap of the data flow, thus the smoothness of throughput during the handoff.

Note that while these metrics — distribution of throughput and throughput gaps — can be used to observe the smoothness of the handoff, these are only QoS metrics using synthetic payloads. To examine the impact to the user experience, QoE metrics must be used. Therefore, Chapter 7 evaluates QoE metrics measured using a *real-life* payload to gain insight into how those QoS characteristics impact the QoE.

Misordering rate and loss rate As discussed in Section 4.6, misordering and loss are presented as ratio. Specifically, misordering rate is a rate of misordering against total number of received packets, and loss rate is a rate of loss against total sent packet counts.

5.3.3 Data collection and analysis

In order to measure the metrics required for the evaluation, `tcpdump` program was used to record the packets as `pcap` files. The packets were collected at the both end-hosts, providing complete picture of the data transmission. By collecting the packets at the end-hosts, testbed did not require any middleboxes for data collection, keeping the testbed network as simple as possible. `tshark` was used to interpret the `pcap` file, providing outputs as comma-separated values (CSV) files. The resulting CSV files were merged, if necessary. The `data.data` column, representing

the most-inner parsed payload part of the packet was trimmed to reduce the storage and memory required for processing. R scripts were used to analyse the CSV files.

Throughput gap Packet *flow* is discrete in its nature. Therefore, between each packets, there exists some amount of gap, also known as inter-packet arrival time (IPAT). The gap was derived from the timestamp of packets received at the receiver host. In this evaluation, a duration between the received packets beyond 0.1 second were deemed a gap in throughput, which are several orders of magnitude higher than the IPAT expected in a 10 Mbps synthetic payload from iPerf2 — for a 10 Mbps iPerf2 User Datagram Protocol (UDP) flow, the IPAT was approx. 1ms. In addition, 0.1 second is well beyond the RTT, which in TCP scenario, can influence the IPAT at the initial phase of data transmission. These gaps were examined per-run as well as cumulatively across all twenty runs per scenario.

Throughput distribution The throughput measurement was taken both cumulatively and in per-run aggregates. For throughput distribution, cumulative throughput measurements were aggregated. For cumulative throughput measurements, packets were grouped by time intervals of 0.5 seconds. IP payload lengths were added within each interval, then divided by the interval duration to determine the throughput during that particular interval. The throughput distribution were examined across twenty runs per scenario. Using R, the interquartile range of the throughput was taken across all twenty runs.

Sequence number analysis — misordering Misordering was determined by whether the sequence number incremented or not (whether that is in the TCP header or for the iPerf2 UDP, in the payload segment). When it decrements, it is either due to, the sequence number wrapping back to the lowest value, or misordering. This particular set of results did not encounter sequence number wrap.

5.4 Results

The primary focus of the evaluation was the overall flow performance with continuous handoffs across multiple networks. The results were collected and aggregated for twenty runs per network

protocol, and transport protocol payload combinations. Specifically, the following have been tested:

Network protocols: ILNPv6, MIPv6

Payload/transport protocols: iPerf2 UDP, iPerf2 TCP

For each combination, twenty runs were used for the results and analyses.

The following sections will discuss the results, comparing ILNPv6 and MIPv6 for each transport protocol used, starting with TCP iPerf2 results. In both transport protocols, no significant misordering was observed.

5.4.1 TCP iPerf2

The results from the TCP runs are shown below. As TCP has flow-control and reliable delivery mechanisms, the results showed the impacts on them from the underlying network protocol data transfer performance.

Example Run Figure 5.2 on page 86 shows plots from a typical run from the runs used in this evaluation with iperf tcp flows. They were captured and observed at the MN. For the MIPv6 run, there were bursts of throughput at the end of each of the handoffs. This was due to the TCP socket queueing and buffering the packets as no ack reaches the sender. See Section 5.5.4 for further discussion regarding the effect. Due to these bursts, MIPv6 did not experience any significant loss. ILNPv6’s smooth handoff with consistent throughput also did not experience any significant loss.

Zero throughput duration — throughput ‘gaps’ As discussed, the handoff timings are different between the MIPv6 and ILNPv6. Since the MIPv6 initiates handoff after losing connectivity, its handoff experiences a duration where no packets can be transmitted. Figure 5.3 shows the cumulative throughput ‘gap’ and the distributions of the each ‘gap’ observed per run. This ‘gap’ in throughput was much larger in the MIPv6 scenario — in a continuous mobility scenario, where handoffs occur constantly (in this evaluation, eight times in total); while ILNPv6 did not experience any, MIPv6 experienced, on average, 55–60 seconds of cumulative throughput ‘gap’

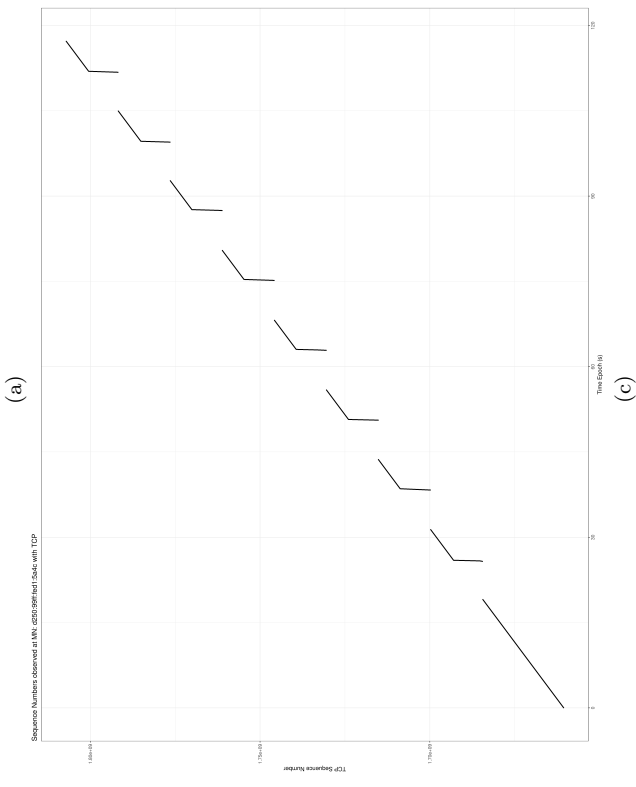
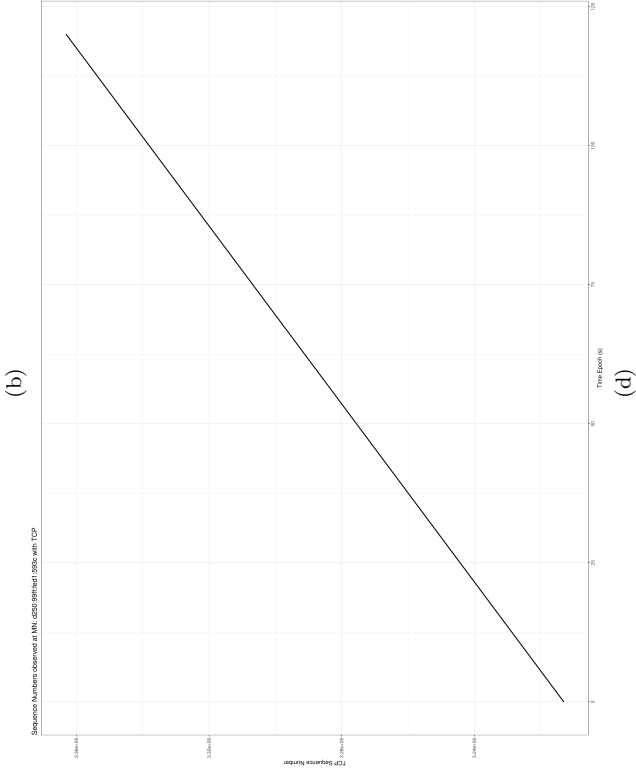
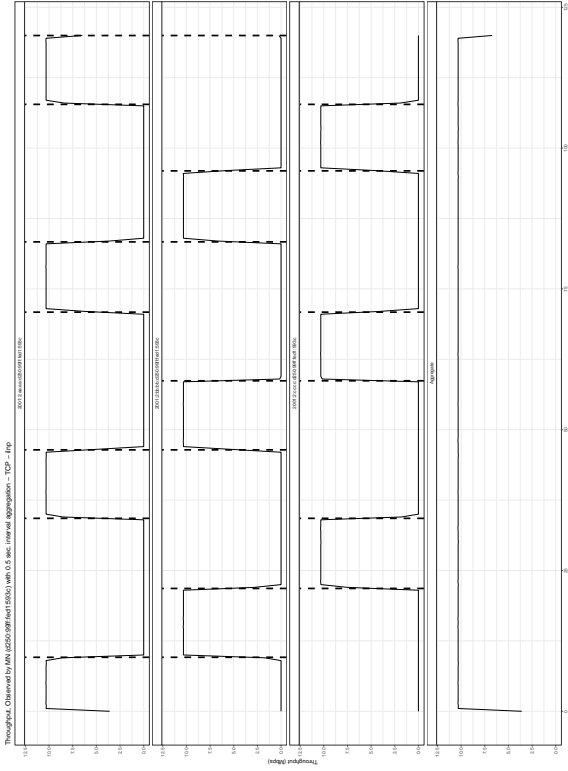


Figure 5.2: The above plots show the throughput and the sequence numbers observed over time in the iperf tcp flow scenario. The throughput is shown in each facet plot: the top three plots in (a) and (b) show throughput received at the MN, at the given prefix/L64, and the bottom plot shows the aggregate throughput. Plots (c) and (d) show the sequence numbers observed over the duration of the flow. The vertical dashed lines indicate mobility signaling packets: LU for ILNPv6; and CoTI, CoT, BU, and BU-Ack for MIPv6. As shown in (a), MIPv6 experienced 'gaps' in data transmission, followed by bursts in throughput. This is also evident in the sequence numbers shown in (c), where the sequence numbers dramatically increased vertically, i.e. almost instantaneously, indicating a burst in transmission. Meanwhile, ILNPv6 experienced a stable, consistent flow of data with consistent increase in sequence numbers.

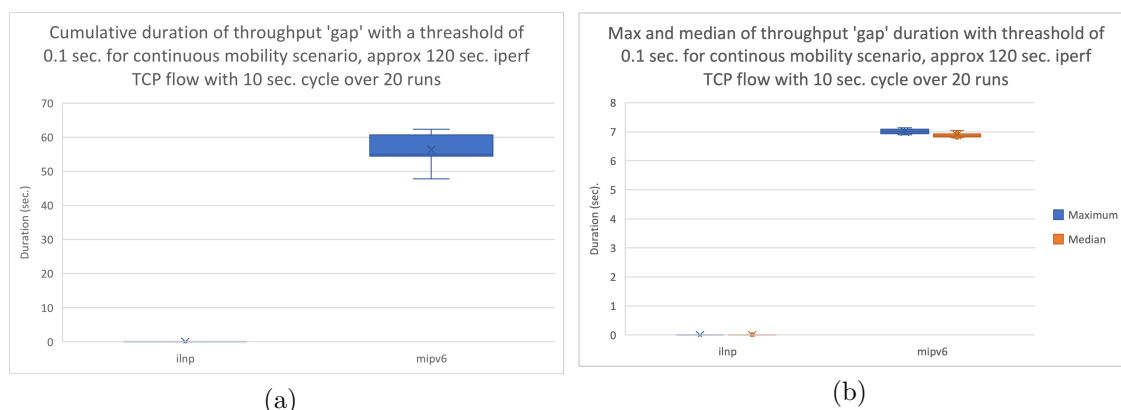


Figure 5.3: The plots show the ‘gaps’ in throughput. (a) shows the cumulative duration of ‘gaps’ in throughput in each 120 seconds flow. (b) shows the maximum and the median of the individual ‘gaps’ observed in 120 seconds flow. The \times indicates the mean.

during the 120 second run. Each individual ‘gap’ observed turned out to be near seven seconds; this duration yielded a loss of approximately 70Mb, 8.8MB of data for 10Mbps flow accumulated over a gap (which was recovered by the aforementioned bursts).

Throughput distribution In the context of evaluating the *stability* of flow, distribution in the observed throughput was used to visualise how consistent the flow was. As shown in Figure 5.4, the throughput observed with ILNPv6 hosts remained stable at the near target throughput of 10 Mbps, while MIPv6 had a much wider distribution. This is directly impacted by the throughput ‘gap’ shown previously in Figure 5.3. Due to the loss of data transmission, throughput fluctuated greatly with MIPv6, while ILNPv6 delivered the consistent throughput at 10 Mbps. The outliers towards 0 Mbps seen with ILNPv6 were observed at the beginning and the end of the communication session. Meanwhile, MIPv6 experienced outliers of high throughput due to the bursts after the loss of throughput during handoff. See Section 5.5.4 for further discussion.

5.4.2 UDP iPerf2

The following shows results from the iPerf2 UDP scenario. The nature of the UDP led to results without the buffering behaviours seen in TCP results. Rather, the effect of 0 throughput duration of MIPv6 led to loss.

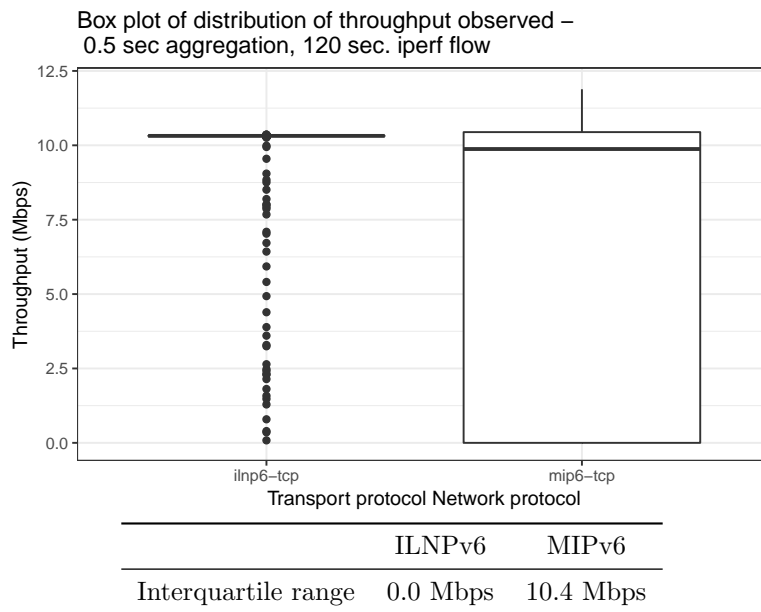


Figure 5.4: The plots show the distribution of observed throughput in 0.5 second intervals for TCP iPerf2 flow. While ILNPv6 experienced most of the throughput at 10 Mbps, MIPv6 experienced a wide range of throughput down to 0 Mbps. Note that MIPv6 TCP results (labelled mip6-tcp) has outliers up to 150 Mbps due to the TCP buffering effects from the testbed, which were omitted for the clarity of the result. The table underneath describes the interquartile range.

Example run Figure 5.5 on page 90 shows plots from a typical run of the runs used in this evaluation with iperf udp flows. As described in the caption, ILNPv6 provided much smoother, continuous flow. UDP does not provide resend or flow-control; the ‘gap’ in throughput directly caused losses, which were visible in the sequence number-time plot.

Throughput distribution Figure 5.6 shows the distributions of throughput observed in 0.5 second intervals. Similar to the TCP scenario, MIPv6 showed a much greater distribution of throughput observed, while ILNPv6 showed a much more stable throughput at 10 Mbps.

Packet loss rate Figure 5.7 shows the packet delivery stats of the UDP iperf flow in the continuous mobility scenario. MIPv6 experienced on average 38% loss, as high as 42% — while ILNPv6 experienced on average about 2% loss with only up to 8%.

5.4.3 Results summary

Overall, the results showed that make-before-break mechanism and much quicker handoff duration of ILNPv6 due to its much simpler messaging provides more consistent and stable flow in both UDP and TCP scenarios. While ILNPv6 was able to provide a continuous data flow during the handoff without any gaps, MIPv6 showed that almost half (55–60 sec.) of the entire duration of the data transfer (120 sec.) was spent waiting for the new path to be usable in both scenarios. In terms of the throughput itself, in 0.5 sec. aggregations, ILNPv6 showed a much more consistent throughput than MIPv6 overall.

5.5 Discussion

5.5.1 IPv6 Router Advertisement in continuous handoff scenario

ILNPv6 uses RA lifetime parameters for its L64 entries in ILCC. RA has two lifetimes:

Preferred lifetime: The lifetime before the prefix becomes not preferred on the host

Valid lifetime: The lifetime before the prefix becomes invalid on the host

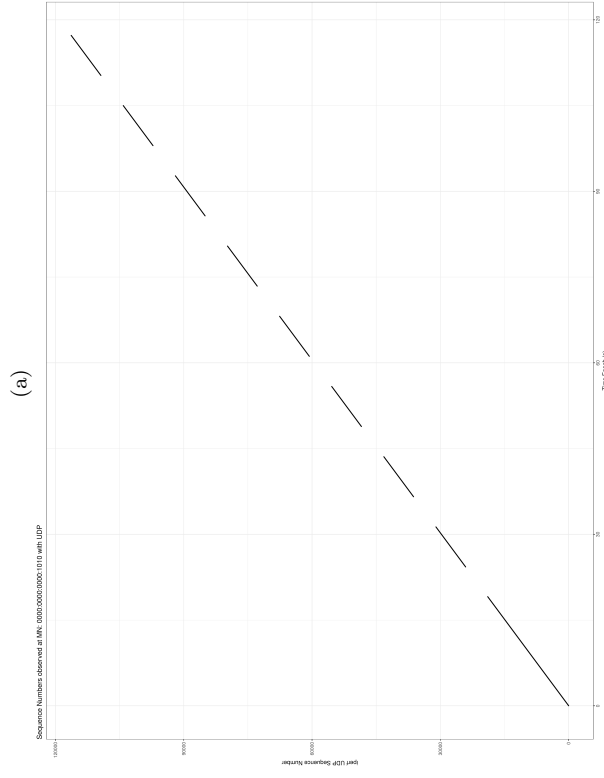
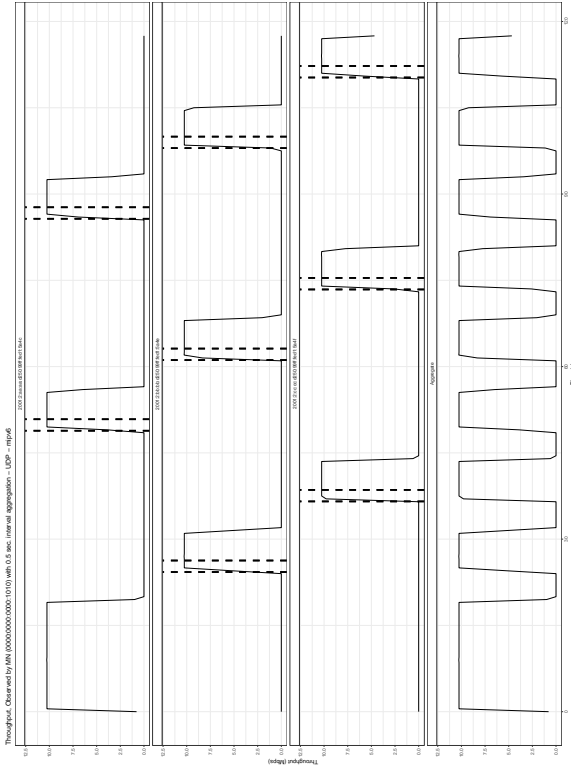
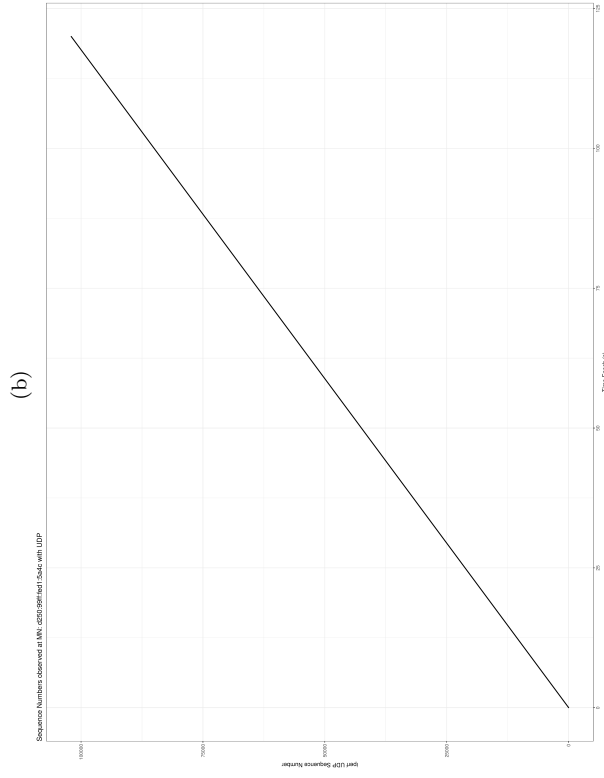
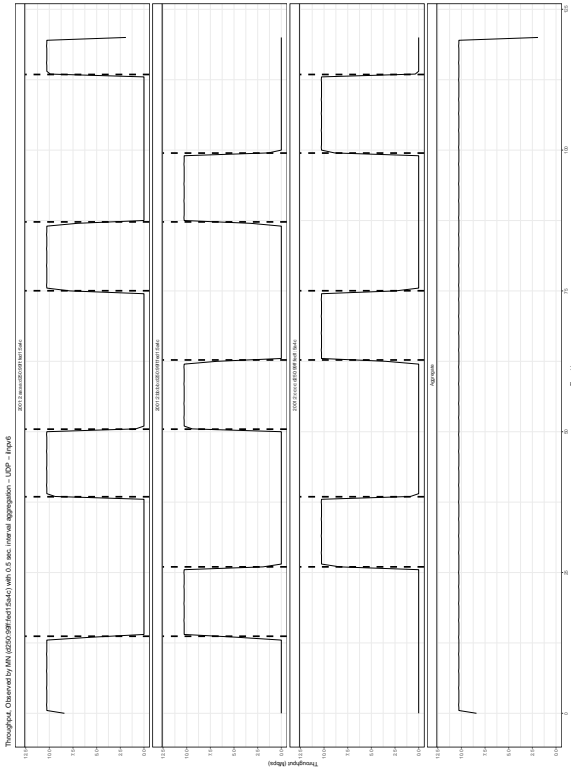


Figure 5.5: Throughput and sequence numbers plotted over time for MIPv6 and ILNPv6 iperf UDP continuous mobility scenarios. The vertical dashed lines indicate mobility signalling packets: LU for ILNPv6; and CoT, CoF, BU, and BU-Ack for MIPv6. Similar to the TCP mobility scenario, MIPv6 experienced significant gaps of throughput during handoffs. As UDP does not buffer or resend packets, loss of throughput during handoff directly led to loss of packets evident in the sequence numbers observed. Meanwhile, ILNPv6 results showed a much more stable aggregate throughput, and no gaps in the sequence numbers delivered.

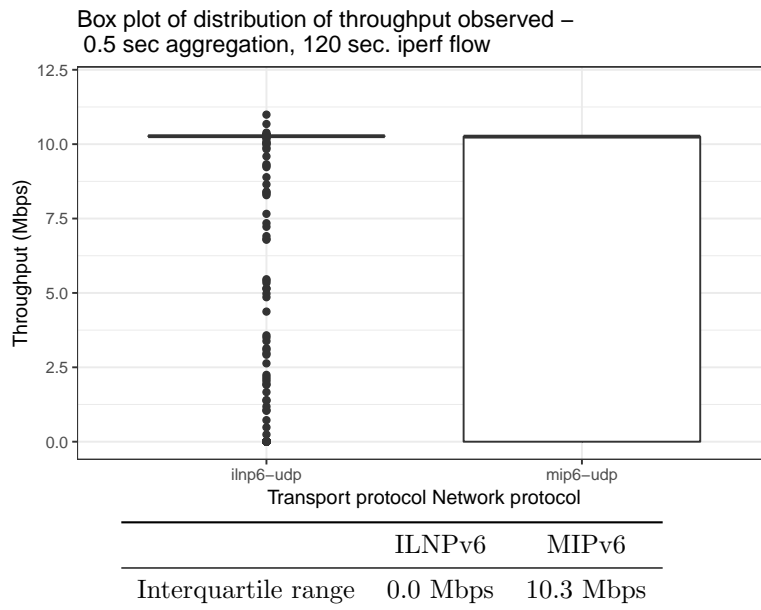


Figure 5.6: The plot showing the distribution of observed throughput for UDP iperf flow. While ILNPv6 achieved stable throughput around 10 Mbps, MIPv6 had a wider distribution extending all the way down to 0 Mbps. As the aggregates were sampled across the whole run, it was evident that ILNPv6 experienced throughput between 0 Mbps to 10 Mbps at the beginning or at the end when some packets spilled over the 0.5 second aggregation buckets. The table below describes the interquartile ranges.

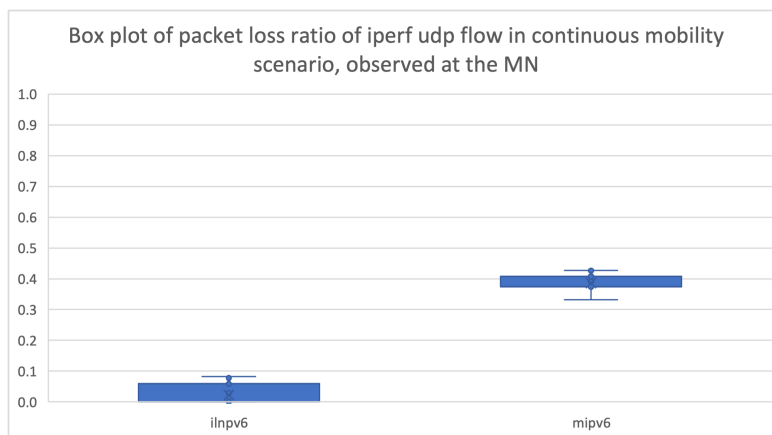


Figure 5.7: A box plot showing the packet loss observed in the iperf UDP flows. Compared to the ILNPv6 flows, MIPv6 suffered a much higher number of packets dropped, approximately 4–6 times more. The × indicates the mean.

In the ILNPv6 context, the preferred lifetime is used as the lifetime of L64. With this implementation, ILNPv6's soft handoffs use RA to trigger handoffs. This fixed-behaviour sets the previous L64 to be VALID and it will not become ACTIVE until it first becomes EXPIRED, then the host receive a new RA after expiry. Therefore, in order to enable continuous handoffs, when returning to a previously used L64, the preferred lifetime must be as short as possible, as the interface remaining VALID will not let a new RA to trigger handoff and set the L64 to become ACTIVE again. The RA lifetimes should be set appropriately to allow the end-hosts to know the reachability and the state of the routers which they are connected to in a more timely fashion. For example, the network may change prefixes; for a host to be aware of this, lifetimes must be short, allowing the hosts to be up-to-date. If the network expects devices to move in and out frequently, the same applies. Naturally, short lifetimes require shorter RA intervals, as infrequent RA may cause the prefix to be not *preferred*, or worse, to become *expired* and invalid. In addition, the RA is an Internet Control Message Protocol v6 (ICMPv6) message, which may be lost at times. Therefore, there should be some tolerance to allow some of them to be dropped before the timer expires. In this testbed, the configuration shown in Chapter 4 Section 4.4.2 was used.

Frequent RA with short lifetimes is also strongly endorsed to be used with MIPv6. `radvd` can set even shorter intervals and lifetimes specifically for MIPv6. However, this is not necessarily adopted to every configuration management mechanism and is not always available as an option. Therefore, this testbed did not opt for those options when configuring the routers, except for when configuring the HA node, which required manually modifying the `radvd.conf` file bypassing the configuration management mechanism present in VyOS. In addition, to make a fair comparison, an assumption was made that none of the access networks are aware of MIPv6 or ILNPv6, but configured to accommodate highly mobile hosts — potentially, this may be done due to the administrator's observation, or the access network may anticipate mobile devices on the network due to the demographic of users it is aiming to support.

In addition, as pointed out in [58], prioritisation of ICMPv6 packets will be important; the loss/delay of ICMPv6 packets such as RA, LU, and BU (and other MIPv6 signalling packets) would impact the handoff performance, especially in continuous mobility scenarios. While it was not experienced in the iperf2 scenarios in this particular experiment, as the throughput was

limited using the iperf2 configuration, the TCP will generally take advantage of the full capacity of the path, potentially causing loss of other packets.

When deciding prefix lifetimes, the RA intervals must be taken into consideration; if the lifetimes are shorter than the interval, the prefix will either become deprecated or invalid. Deprecated prefixes are ‘discouraged’ prefixes, which means no new communication should use them but packets to those prefixes should still be received. RFC2462 [63] states that deprecated prefixes may be used for communications where switching prefixes may cause an issue, such as ongoing TCP communication. Invalid prefixes are prefixes that ‘should not be used’ [63]. Therefore, the consequences of improper prefix-lifetimes and RA intervals can be severe.

In highly mobile scenarios, the goal of RA is to provide the most up-to-date prefix information and indication that the router is available. However, since ICMPv6 can be lost depending on the network condition, there should be some ‘buffer’ allowing small amounts of loss, i.e. a single RA loss should not lead to false prefix-expiry when the router is still connected and operational.

As stated in RFC4861 [64], the minimum interval must be no less than 3 seconds and no greater than $0.75 \times$ maximum interval; and the maximum must be no less than 4 seconds and no greater than 1800 seconds. In the evaluation testbed, packet loss was rather rare, both under load with the test scenarios and without. Therefore, the interval was set as follows: preferred-lifetime at three times the minimum interval, valid-lifetime at three times the maximum interval; where the minimum interval was set to 3 seconds, the maximum interval was set to 5 seconds, the preferred-lifetime was set to 9 seconds, and the valid-lifetime was set to 15 seconds.

5.5.2 Wireless versus wired technology

The experimental scenario used was an emulation of the user mobility scenario described in Chapter 4 Section 4.1. As described in the user mobility scenario, the mobile device will be using wireless technologies. However, the experimental scenario used wired Ethernet connectivity. This was in order to improve the reproducibility of the results. First, 802.11 wireless association times are non deterministic, impacting the handoff latency (see Section 5.5.3 for further details). By using wired technology, external factors such as radio interference was eliminated; in order to eliminate radio interference with wireless technology, it would require a substantial infrastructure

such as a Faraday cage, which was not available and was unrealistic to mandate. Another hurdle in using wireless technologies was the licensing; in order to correctly carry out wireless experiments involving cellular technologies such as 4G and 5G, appropriate licensing and large infrastructures enabling them would be required.

While the wireless technologies were not used, the fundamental focus of the mechanism is the network layer mobility. That is, the focus of the work is solving the *inter-workings* of different technologies in the mobility context. The data plane and control plane will remain the same regardless of the physical technologies or media access control (MAC) protocols used. In fact, handoffs within the same layer 2 MAC domain will still be needed, as network layer handoff will not take place. Any differences between technologies will likely be QoS — which upper layer technologies are already capable of adapting to — and the delay for link establishment, which greatly depends on individual technologies. Therefore, wired Ethernet was sufficient to show the effectiveness of the ILNPv6 identifier-locator split architecture and its mobility mechanism in continuous mobility scenarios.

5.5.3 Handoff latency

There are several factors contributing to the duration required for handoffs as outlined in Section 4.5.5. However, we do not expect the ILNPv6 LU handling to be the bottleneck in handoff latency. By design, ILNPv6 LU is 1 RTT — MN sends an LU, and then CN responds with an LU-Ack. One factor impacting the handoff latency is layer 2 signalling and delays associated with it. At layer 2, various technologies require different procedures to establish a link. 802.11 network association time is non-deterministic, and the security mechanism would require a security handshake in addition to establishing a network association with the base station. Another factor is at the layer 3, with IPv6 RA, which ILNPv6 uses to initiate a handoff. In the host configuration, the end-hosts are configured to send RS as soon as possible. However, RA response is not necessarily instantaneous; RFC4861 [64] Section 6.2.6 states that the router must 1. rate-limit RA response, and 2. delay RA response by ‘a random time between 0 and `MAX_RA_DELAY_TIME`³ seconds.’ In addition, even after the prefix is received, regardless of the

³RFC4861 section 10 defines it as 0.5 seconds.

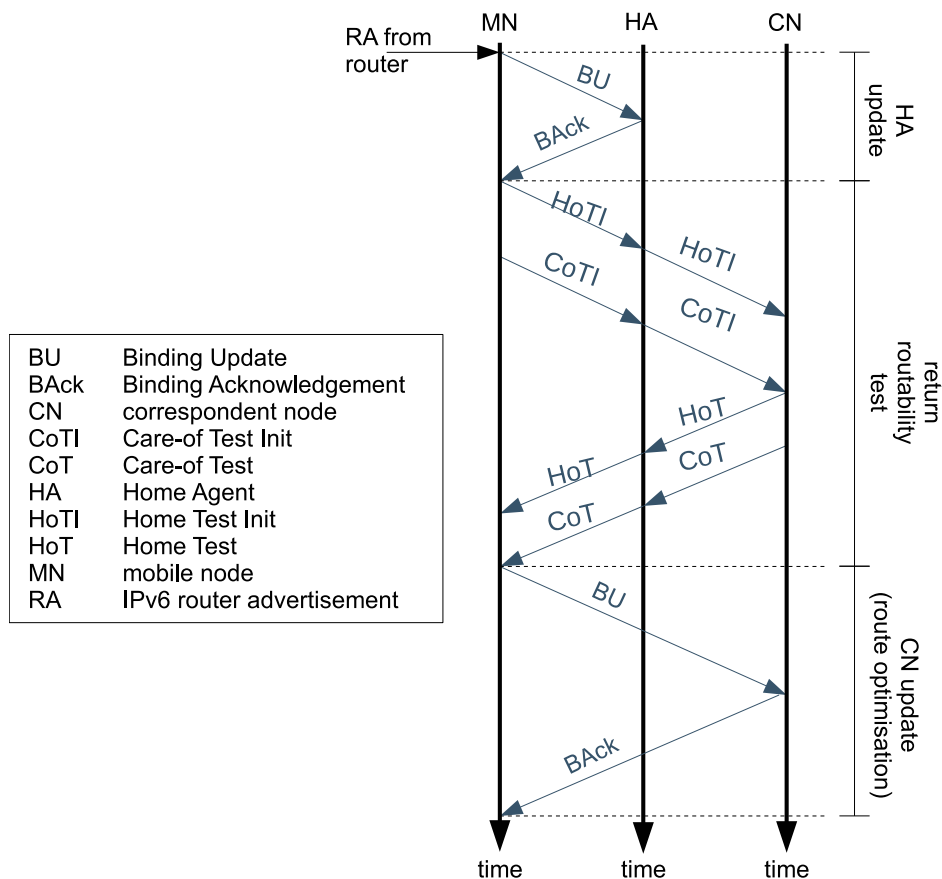


Figure 5.8: A handoff message timeline for MIPv6 with route optimisation. The handoff is triggered by a loss of communication. The handoff requires well beyond 1 RTT, due to the number of messages required.

network layer protocol, DAD or optimistic DAD must occur, which will also be a factor. Provided that there is no LU loss, the lowest handoff latency possible for LU is approximately 1 RTT — this is between the MN and the CN unlike MIPv6, which involves HA in addition to CN, adding further latency to its signalling. Moreover, comparatively, ILNPv6 requires significantly less signalling than MIPv6; as shown in Figure 5.8, MIPv6 requires at least three times more messages, some of which has to go through the HA.

5.5.4 Buffering effect in TCP scenario

During the handoff using MIPv6, since it does not employ a make-before-break mechanism, the data transfer completely halts when the ‘previous’ connection becomes unavailable to the host. In such a scenario, as the TCP acknowledgement will also halt, the host will buffer the segments to send while the path is not available. Upon resuming reachability to the destination, TCP will send bursts of buffered packets in order, at a much more rapid pace, outside of iperf’s ability to control. iperf’s ‘throughput control’ is done before TCP’s socket queue or congestion control mechanisms take effect and does not have any knowledge of link-state, nor considers the window size or the state of queue. Since the testbed does not restrict the throughput (iperf2 itself attempts to pace the output), in this environment, it reaches near 80Mbps in those phases; however, in a more realistic environment where a throughput constraint on the path exists, packets will not be transmitted at such rapid rates, leading to delayed delivery of packets or potentially, misordering and loss. Media applications, such as video on demand, may face issues such as depleting the buffer during handoff, or causing buffering mechanisms to require a larger buffer, and potentially, leading to some inconsistency in playback performance. This depends on how the streaming mechanisms handle the delayed delivery and inconsistency in throughput during the handoff. More time sensitive applications will be more greatly affected depending on the duration of the zero-throughput duration. In those applications, resent packets may well be a waste.

The testbed network was kept simple for this study; however, if the testbed network included some mechanisms to hard-limit the throughput, which would emulate a more realistic typical network path condition over the Internet, such peaks of high throughput would not occur, leading to a different behaviour, which would lead to a much worse results for MIPv6. One way to create such a throughput bottleneck in the testbed may be to limit the link speed of the appropriate paths in the network, or by using some QoS control mechanisms. While built-in QoS control within the router or netem could be used, the aim here was to use commercial routers in simple configurations.

5.5.5 Round trip time (RTT) emulation

This particular evaluation did not introduce any additional delays in RTT. However in the ‘real world’, it is possible, if not common, for mobile devices to encounter varying RTT to the correspondent node, depending on the topology of the respective network they are attached to. While introduction of heterogeneous RTT may make the scenario more similar to the real-world, the aggregation and the analysis of multiple handoffs in a single run may have been more challenging. As an isolated evaluation of continuous handoff, having a homogeneous RTT made sense.

Significant changes to RTT are unlikely with handoffs across the same layer 2 technology; however, with a vertical handoff, it is more likely to occur as RTTs are impacted by network topology, layer 2 protocol characteristics, and physical medium characteristics. Relative to horizontal handoffs, vertical handoffs are more likely to exhibit a drastic change to the end-to-end path. With the heterogeneous RTT on the path, while impact on the UDP scenario will likely be limited to misordering of the UDP packets and network layer handoff signalling packets, the impact on TCP will be much more severe. One of the impacts will be misordering being treated as loss, causing re-transmissions and potentially further delaying the packet delivery. This will also impact the congestion window (CWND), restricting the throughput. Depending on the size of added latency on the path, retransmission timeout (RTO) may occur, also causing a slow-start. All of those TCP behavioural impacts are possible, the scale of the impact hugely depends on the combination of the RTT changes that the host encounters. Handoffs from a path with short RTT to a path with long RTT will likely cause the above behaviours, while the opposite will likely have minimal impact.

In fact, having heterogeneous RTT would be a separate evaluation for a very particular scenario — whether that is a certain combination of networks to perform the handoff from/to and in certain order. This evaluation, instead, was an evaluation of the protocol performance, in the context of a general solution to mobility in continuous movement. Therefore, each handoff was kept the same to purely observe the handoff performance of each solution.

5.5.6 Impact of interface selection mechanisms using route metric

This particular implementation of ILNPv6, implemented on Linux 4.9 LTS kernel, based on [58], utilised the existing IPv6 routing selection mechanism's route metric to select the 'newer' interface that was added more recently. One of the advantages of this design was minimising modifications to the existing Linux packet processing mechanisms and routing selection. However, this separated the interface selection and L64 selection mechanisms. One of the side-effects of this mechanism was that the L64 selection may not necessarily follow which interface it may or may not leave from. In Linux network packet processing, since both the source and the destination address are required to be set at the transport layer packet processing phase due to the transport layer state tuple and checksums, the route and the interface are decided at a much earlier phase than the network layer packet processing phase. Therefore, in some situations where a route has been picked before the 'new' interface receives RA, some packets may be prepared to leave the 'old' interface, but when they reach the IPv6 and ILNPv6 processing path, the new interface and its L64 may be available and their source L64 overridden at the network layer. Such event leads to a small number of packets with the new L64 leaving from the 'old' interface. While in this testbed, that did not lead to packet loss, if the network enables source address validation, it may lead to those packets being dropped.

5.5.7 Omitted L64 state — AGED

As discussed in Section 5.1.4, AGED was omitted from this implementation of ILNPv6. The potential consequence of not implementing AGED will be apparent in situations where ICMPv6 packets are likely to be lost. Specifically, in this particular testbed's routers' RA configurations, the L64 will become AGED when three RAs are lost for that particular L64. For example, the network may be under stress, which could lead to a small amount of loss. With a short lifetime, if multiple RAs are lost, the prefix may become deprecated and the L64 to become EXPIRED without the presence of AGED state. While the RAs may be lost, it is possible that the network could still be operational to some extent. In such situation, the expiry of L64 can disrupt the communication; but if the AGED state is implemented, the host could potentially still hold the communication.

However, in reality, this particular scenario is rather rare and is more likely that the network is severely disrupted in a scenario where RA is disrupted. If the RAs are lost, it indicates that there is a catastrophic disruption to the network or the router and therefore the communication itself would be disrupted too. In such events, applications should be made aware of the issue sooner rather than waiting for packets that may not arrive reliably.

5.6 Summary

In this chapter, the results showed that ILNPv6 is able to deliver consistent throughput in a true continuous mobility scenario at the network layer. That is, ILNPv6 can enable continuous mobility for any transport protocol across any lower layer technologies, without loss of throughput during handoff, providing a smooth handoff. Comparison with MIPv6, which is the current IETF mobility solution, showed that ILNPv6 provided much more consistent data flow during the handoff for both the UDP and TCP — while MIPv6 caused gaps in throughput during the handoff, disturbing the flow of data between the hosts. ILNPv6 provides smooth handoffs for a completely agile node. In this evaluation, ILNPv6 outperformed MIPv6 in every metric measured: throughput, loss, and handoff performance.

Chapter 6

Mobility and Multihoming

Duality

In the modern mobility context, devices commonly have more than one connectivity available at a given point. For example, a smartphone may be associated with an 802.11 network at a site while also connected to a cellular base station. In such a scenario, the smartphone might only be able to utilise the 802.11 network, not making use of the cellular connectivity. Current Internet Protocol (IP) does not allow applications to utilise multiple connectivities without implementing special mechanisms to manage them. As Identifier Locator Network Protocol v6 (ILNPv6) decouples the locator and the identifier of the host, multihoming is simply using additional locator(s) to/from the multihomed host.

Multihoming in ILNPv6 can be seen as an enhanced version of the soft-handoff. During the soft-handoff, the ILNPv6 mobile node has two interfaces connected to two distinct networks, where one is set to ACTIVE and other is set to VALID. Specifically, the interface that will be disabled later, is set to VALID, while the ‘newer’ interface is set to ACTIVE once the LU is sent to the Correspondent Node (CN). As different paths may have different delays, it is possible that the packets sent prior to the Locator Update (LU) packet may reach after Locator Update Acknowledgement (LU-Ack) arrives at the Mobile Node (MN) on the interface that is planned to be disabled. In such a scenario, as long as the previous interface is VALID and the path to

the interface is still available, those ‘late’ packets will still be delivered and received. This is a special form of multihoming, whereby a mobile host can receive on the two distinct interfaces with its respective locators during a handoff, with the intention to only use one after the handoff. Taking this mechanism further to enable both sending and receiving from multiple interfaces with multiple locators enables true multihoming.

The previous chapter presented a truly mobile node continuously moving across different network boundaries. When combined with continuous mobility, multihoming can enable true ubiquitous communication, whereby a host could utilise any combination of connectivities available to it dynamically. This is different to the existing multihoming solutions often seen in site-multihoming, where a site has a fixed set of networks available to it.

Currently, there are no network-layer mechanisms that enable mobility and multihoming simultaneously as a duality. Mechanisms such as MultiPath-TCP (MP-TCP) and QUIC have specifications/mechanisms defined for both mobility and multihoming; however, these are:

1. transport layer only solutions
2. not designed with both mobility and multihoming as the primary functions

Since these are transport layer solutions, no new transport layer protocols or legacy applications can take advantage of them, thus those need to come up with a solution of its own. Solutions like Host Identity Protocol (HIP) require application changes, which requires new applications to specifically implement HIP, and existing applications will not be able to make use of them. Since ILNPv6 is a purely network layer solution that does not require application binaries to change, existing applications can take advantage of ILNPv6’s host mobility-multihoming mechanism without any changes. Therefore, in this evaluation, the mobility-multihoming scenario will be evaluated solely with ILNPv6 only, and data will be collected and analysed to observe its behaviour with different payloads and round trip time (RTT) configurations.

In this chapter, both mobility and multihoming are implemented as a duality and an empirical evaluation is conducted with continuous multihomed-mobility scenarios.

6.1 Overview of mobility-multihoming duality mechanism in ILNPv6

The following describes how the mobility-multihoming duality mechanism functions in ILNPv6 with proof-of-concept host-wide load-balancing. Previous implementations always had one ACTIVE Locator (L64) at the MN, but not more than one at any given point. During the ‘overlap period’, the MN has the ‘previous’ L64 state as VALID, accepting packets with the ‘previous’ L64 as the destination but no new packets are sent, while the ‘new’ L64 state is ACTIVE, which would actively be transmitting and receiving. However, with mobility-multihoming duality, multiple L64s will be ACTIVE, allowing both sending and receiving from all ACTIVE L64s. This proof-of-concept implementation includes a host-wide load-balancing mechanism, using a simple deficit round-robin (DRR) [70] mechanism on both source and destination L64 selection, allowing both send and receive to happen on the ACTIVE L64s present on the MN. In addition, this includes a mechanism to disable network interfaces from the communication on the host by listing them in the disabled_interface list. This mechanism gracefully removes the L64s associated with those interfaces listed on the disabled_interface list; this is done by notifying the correspondent nodes with the new set of L64s, allowing those interfaces to be disconnected without loss of packets and disruption to the communication.

The following describes how each stage is performed:

A. Adding connectivity

In an ongoing communication between two hosts, the ILNP Communication Cache (ILCC) holds one Node Identifier (NID) and one L64. Upon adding a new connectivity on the MN, the following steps take place:

1. The layer 2 link-up occurs at the new interface on the MN.
2. An IPv6 router advertisement (RA) is received at the MN.
3. ILNPv6 adds the new prefix as the L64 in the local ILCC in the MN, and sets the new L64 as VALID.
4. The MN sends an LU with two L64s: one new L64 and the one that is currently in use.
5. The MN updates the new L64 as ACTIVE, and begins sending from it.

6. The CN receives the LU from the known L64, and updates its ILCC entry of the MN with the additional L64.
7. The CN sends an LU-Ack to the MN's newer L64, acknowledging its use, and begins sending to the MN's new L64.
8. The MN receives the LU-Ack.

B. Removing connectivity

With an ongoing communication utilising multiple connectivities between two hosts, the following steps take place to gracefully remove a connectivity.

1. On the MN, using the ILNPv6 sysctl interface entry, add an interface to be removed from the communication to the disabled interface list.
2. The MN identifies the L64 belonging to the interface and sets its state to VALID.
3. The MN sends an LU with all ACTIVE L64s, that is without the L64 that belongs to the interface set in the disabled interface list in step B1.
4. The CN receives the LU with one less L64, and sets the state of the removed L64 as VALID in the ILCC entry for the MN.
5. The CN sends an LU-Ack to the MN's L64 that is included in the LU.
6. The MN receives the LU-Ack. The MN can now turn off the interface.

By combining the two sets of actions, ILNPv6 hosts can dynamically add and remove connectivities without disruption to the data communication, allowing the host to move completely away from the initial connectivity, and then back, realising completely agile and flexible network connectivity between the two hosts. Figure 6.1 shows an example of a mobility-multihoming duality scenario.

6.2 Changes to ILNPv6 and Linux kernel to enable mobility and multihoming duality

In order to enable multihoming, ILNPv6 implementation was modified. In addition, parts of the existing User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) packet

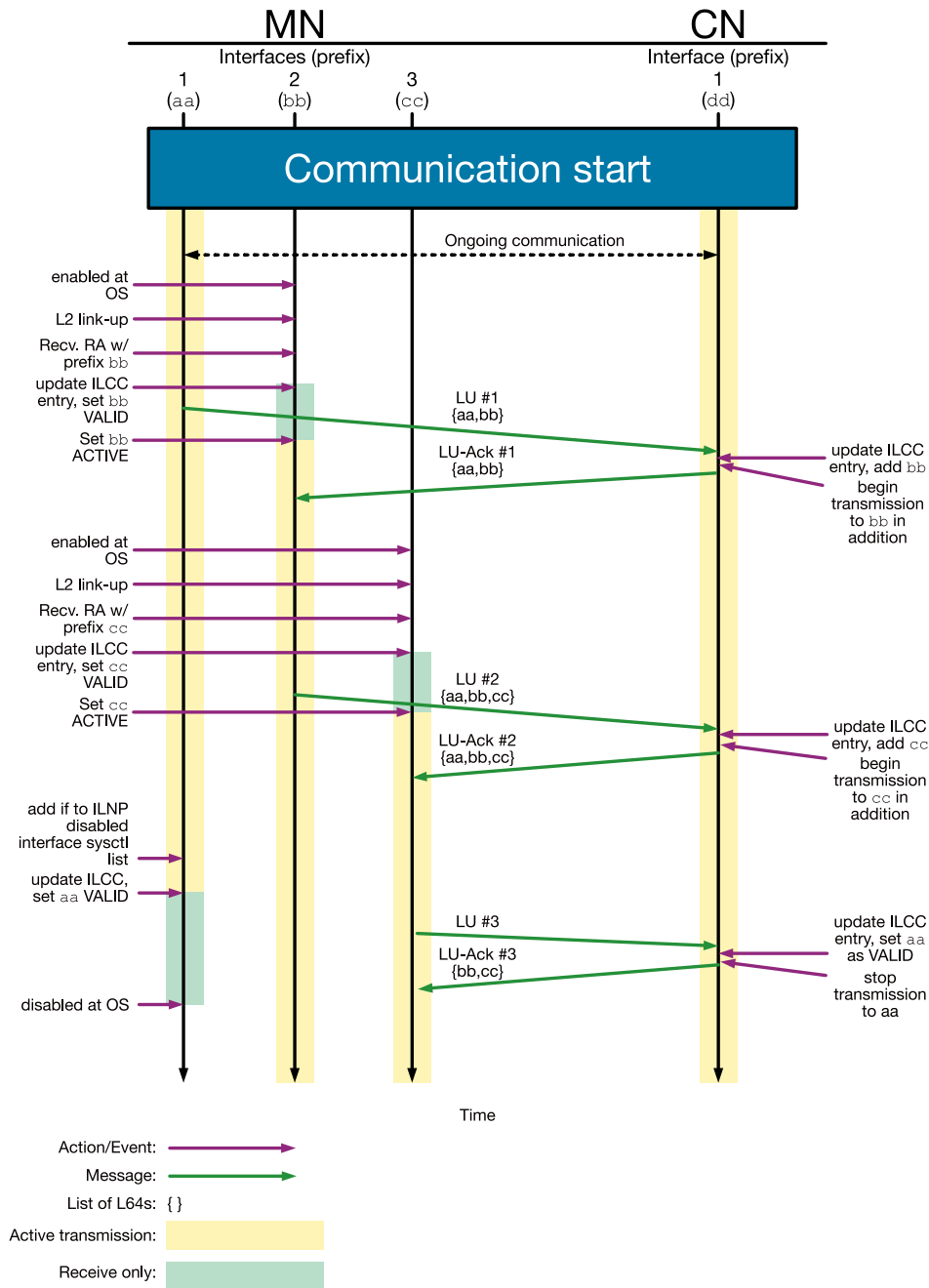


Figure 6.1: A timeline diagram showing an example of mobility-multihoming duality scenario. The MN has three interfaces and the CN has one interface. The MN and the CN begin a communication session using the single interface on both sides. The MN activates interface 2, receiving prefix bb, sending an LU and setting the new locator as ACTIVE. The CN responds with an LU-Ack accordingly, acknowledging the new locator set. The MN continues to activate another interface, which is handled accordingly by the CN as well. The MN then lists the first interface in the `ilnp_disabled_interface sysctl` list, triggering another LU and state changes in the ILCC. After the CN acknowledges the removal of the first interface, the MN removes the interface.

processing mechanisms required modification to allow multiple L64s to be used; specifically, the route lookup mechanism had to be modified for the purpose of facilitating the source L64 selection mechanism. The destination L64 selection only requires changes in the existing destination L64 selection mechanism.

The changes required are:

1. Modifying the route selection mechanisms
2. Implementing a simple load-sharing mechanism
3. ILNPv6 LU message modification
4. Additional ILNPv6 LU code values implementation
5. ILNPv6 ILCC Locator status management and RA handling during multihoming
6. An additional ILNPv6 sysctl interface to enable/disable a locator
7. An additional ILNPv6 sysctl value to change the ILNPv6 host-wide operation mode

The following sections describe them in further detail.

6.2.1 Route selection mechanism and transport layer packet processing

The previous continuous mobility mechanism used the route metrics parameter to change the outgoing interface and the next hop to prioritise the route associated with the most recently configured interface. The route metric-based mechanism provided immediate change to the route without requiring modifications to the flow lookup process in the outgoing packet processing mechanisms. The flow lookup process is executed from distinct processing code paths for UDP and TCP, specifically during the ‘corking’ of the packet and preparation of the `sock_buff` struct. However, such a mechanism does not allow multipath transport.

First, the route metric-based mechanism is less fine-grained and inflexible — as soon as a lower-metric route is added, the entire host’s outgoing route changes; however, it is not possible to control the precise timing of its change and it has no relation to the individual packets in the packet processing path. In order for ILNPv6 to select a source L64 for multipath transport, the outgoing interface must correspond to the L64 selected by ILNPv6 on a per-packet basis; however,

the two distinct transport layer packet processing mechanisms select the outgoing interface at the early phase of their packet processing. In the previous continuous mobility implementation, this occurred before ILNPv6 interacted with the outgoing packets. This phase of packet processing had to be modified in order for ILNPv6 to gain the ability to select the outgoing interface and the route, such that the selected source L64 is reflected accordingly.

The packet processing path involves route lookup when UDP and TCP packet headers are being constructed before they are passed onto the network layer packet processing mechanisms. This is to ensure that the pseudo-header source address (and checksum calculated using it) matches with the actual source address and the outgoing interface. In the context of conventional IP, the actual source address and the respective outgoing interface do not change; therefore, they are cached after the initial lookup. However, that is not the case with ILNPv6, especially in the context of multihoming, where every other packet may leave from distinct interfaces.

To enable multipath transport, specifically, to allow ILNPv6 to select the outgoing interface, ILNPv6 needs full control of the route selection mechanism. In order to realise this, both the UDP and TCP packet processing mechanisms were modified with additional `ilnp6_src_l64_select()` function to correctly set the outgoing route and the respective interface. Specifically, by preventing the caching of routes and triggering route lookup for every packet, the additional ILNPv6 function gained the ability to control each packet accordingly. Figure 6.2 shows a simplified diagram illustrating how UDP/TCP packet processing was modified to enable source L64 selection and respective outgoing interface selection.

The call graphs in the following Section 6.2.1 provide further detail as to how ILNPv6 and Linux kernel were modified to enable this mechanism.

Call graphs of packet processing paths

To give full control of outgoing route to ILNPv6, new function was added and additional functions required modifications as summarised in the callgraphs shown in this section. Figure 6.3 shows callgraph of UDP ingress (i.e. incoming) packet processing path, outlining changes to the existing Linux kernel UDP packet processing calls. Figure 6.4 shows callgraph of UDP egress (i.e. outgoing) packet processing path, outlining changes to the existing Linux kernel UDP packet processing calls. Figure 6.5 shows callgraph of TCP ingress (i.e. incoming) packet processing

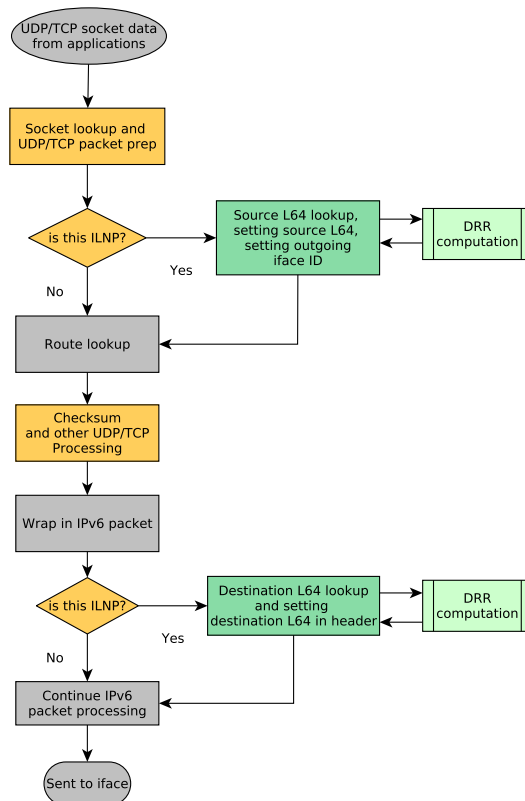


Figure 6.2: A flowchart describing UDP/TCP packet processing with ILNPv6 mobility-multihoming duality mechanism with Deficit Round-Robin (DRR) load-sharing. Boxes coloured grey indicate unmodified processes. Yellow boxes indicate modifications. Green boxes are the additional mechanism in `ilnp6.c`.

path, outlining changes to the existing Linux kernel TCP packet processing calls. Figure 6.6 shows callgraph of TCP egress (i.e. outgoing) packet processing path, outlining changes to the existing Linux kernel TCP packet processing calls. As TCP is a much more complex protocol, the processing mechanism required more changes than UDP.

6.2.2 Load-sharing

Load-sharing is an application of host multihoming, where the throughput of more than one interface available to the host is aggregated and the traffic is shared amongst them. To evaluate the multihoming mechanisms, this implementation of multihoming-mobility in ILNPv6 was implemented with a fixed-behaviour where the load is shared equally to all available interfaces at any given moment. This is achieved by implementing a simple deficit round-robin (DRR) [70] mechanism to select the next outgoing source and destination L64s of the packets. The ILCC L64 entry stores the DRR score derived from the bytes sent from each packet's packet length, which is extracted from `struct sock_buff` data structure. Each time an additional interface is added, all of the DRR score entries are reset. For both the source L64 and the destination L64, the one with the lowest score at any given time is selected. As each L64 is selected for outgoing packets, the DRR score on the selected L64 is increased, while the others are decreased.

6.2.3 Extension to the Locator Update message in ILNPv6 implementation

LU messages are used for ILNPv6 hosts to communicate the changes of their use of locators to the corresponding hosts. Previously, in the mobility context, only one locator was required in the LU, as only one locator was active at any given point; therefore, the previous implementation with mobility implemented it as such. However, in multihoming scenarios, multiple locators must be active, and thus the LU must be able to carry multiple locators. The locator update message implementation was extended to enable multiple L64s to be sent in accordance to RFC6743 [3], shown in Figure 6.7, with some modifications described in Section 6.2.4. Therefore, to allow the handling of multiple locator entries in LUs, changes were required in `ilnp6.c` and `ilnp6.h` to change the LU message handling and structure definition.

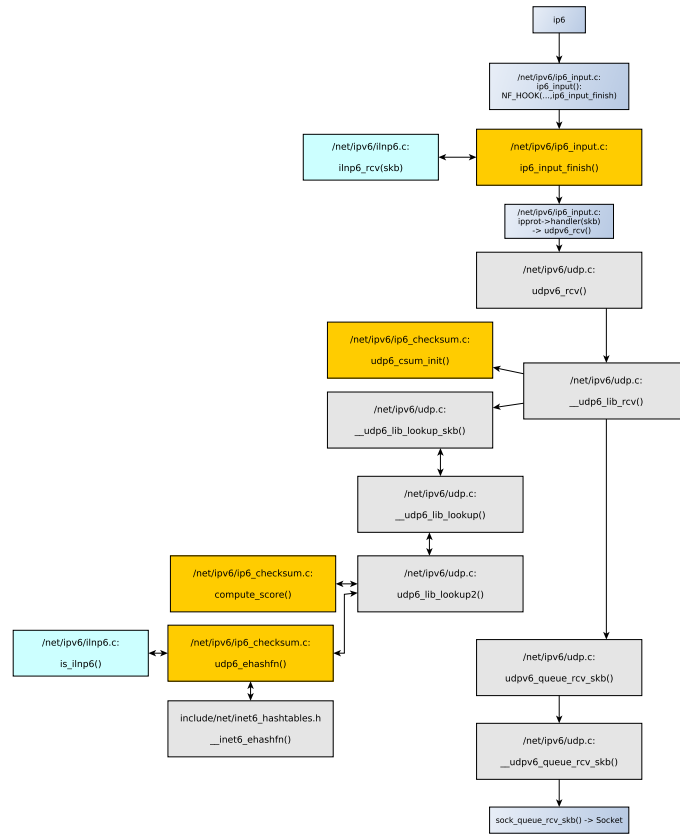


Figure 6.3: A callgraph of the UDP ingress processing path. Grey boxes indicate **unmodified** functions, yellow boxes indicate **modified existing** functions, blue-grey boxes indicate **references to the functions** such as pointers, cyan boxes indicate **ILNPv6-specific functions in the ILNPv6 file**, and boxes with a dotted outline indicate **new functions** for implementing the mobility-multihoming duality mechanism.

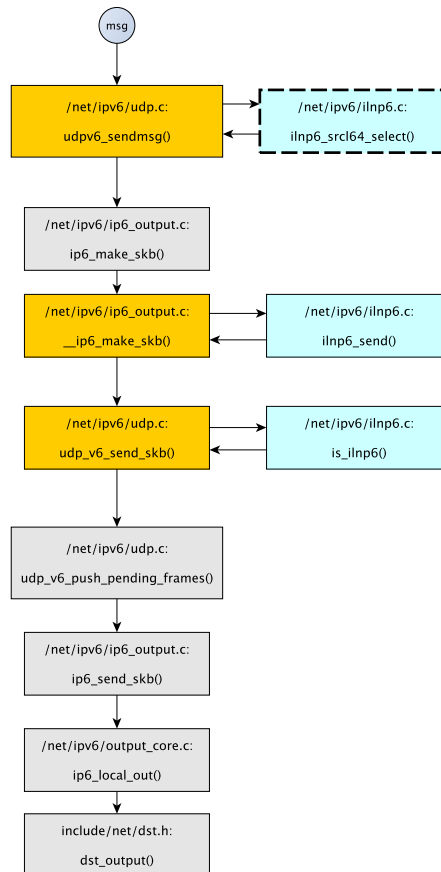


Figure 6.4: A callgraph of the UDP egress processing path. Grey boxes indicate **unmodified** functions, yellow boxes indicate **modified existing** functions, blue-grey boxes indicate **references to the functions** such as pointers, cyan boxes indicate **ILNPv6-specific functions in the ILNPv6 file**, and boxes with a dotted outline indicate **new functions** for implementing the mobility-multihoming duality mechanism.

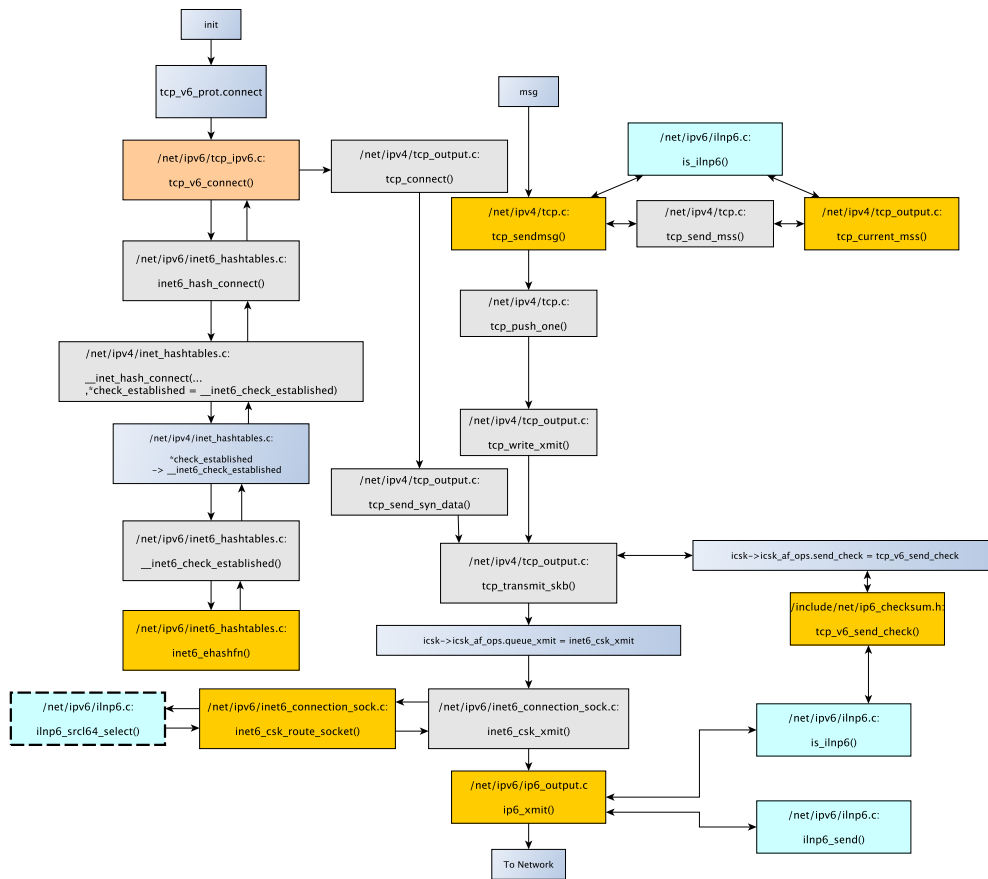


Figure 6.6: A callgraph of the TCP egress processing path. Grey boxes indicate **unmodified** functions, yellow boxes indicate **modified existing** functions, blue-grey boxes indicate **references to the functions** such as pointers, cyan boxes indicate **ILNPv6-specific functions in the ILNPv6 file**, and boxes with a dotted outline indicate **new functions** for implementing the mobility-multihoming duality mechanism.

ICMPv6 Locator Update message

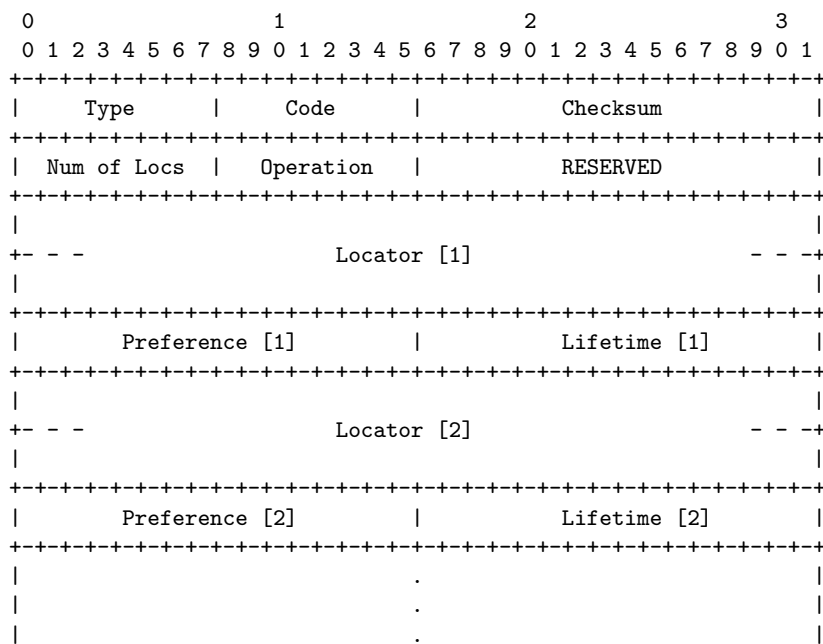


Figure 6.7: The ILNPv6 LU message structure based on the message format from RFC6743 [3].

“Code” value	“Code” name	Description
0	ILNP_LU_NONE	Defined in RFC6743[3] and RFC6754[71]. Backwards compatibility only — must not be used but should be accepted for incoming message.
1	ILNP_LU_UNUSED	This value MUST NOT be used.
2	ILNP_LU_ADD_HARD	“Add locator(s) — hard update”. Locator(s) must be added to the set and enabled immediately.
3	ILNP_LU_ADD_SOFT	“Add locator(s) — soft update”. Locator(s) must be added to the set in soft update manner.
4	ILNP_LU_DELETE_HARD	“Delete Locator(s) — hard update”. Locator(s) must be deleted from the set immediately.
5	ILNP_LU_DELETE_SOFT	“Delete Locator(s) — soft update”. Locator(s) must be deleted from the set in soft update manner.
6	ILNP_LU_REPLACE_HARD	“Replace Locator Set — hard update”. Locator(s) must be replaced immediately.
7	ILNP_LU_REPLACE_SOFT	“Replace Locator Set — soft update”. Locator(s) must be replaced in soft update manner.

The “hard” and “soft” updates differ in how the two hosts, the sender and the receiver of the LU message, handle the update. Generally, with “hard” updates, the changes are applied immediately without need for the acknowledgements from the CN. Contrarily, with “soft” updates, the changes are applied in stages; that is, at the sender of LU, there is an additional behaviour between when the host sends the LU and receives the LU-acknowledgement.

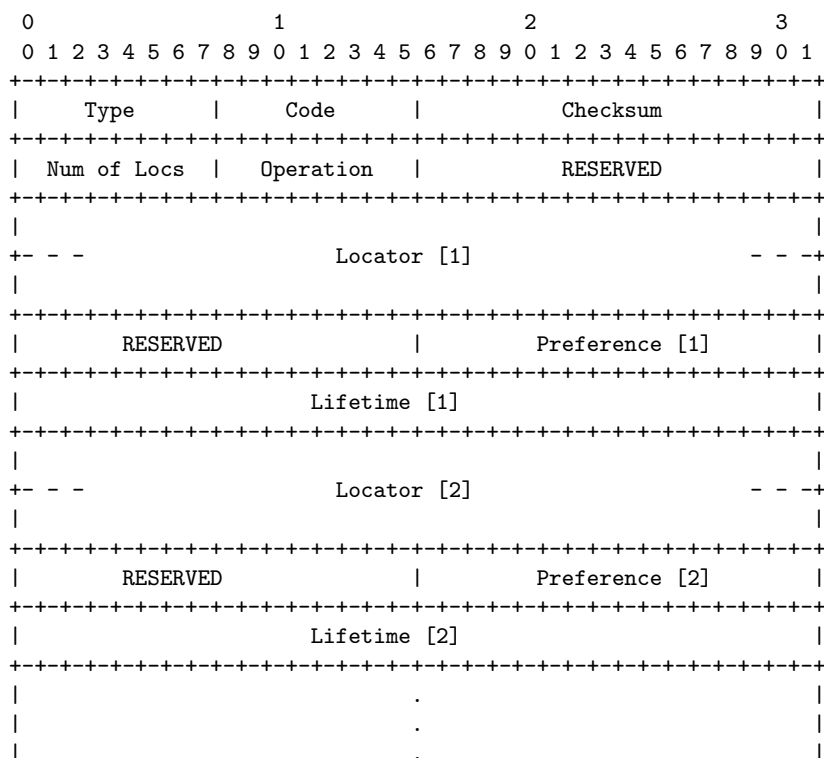
Table 6.1: A table describing the new updated Locator Update codes currently in the draft RFC.

6.2.4 New Locator Update message in draft RFC

The existing RFC6743 [3] specifies the LU messages with multiple L64s as shown in Section 6.2.3 in Figure 6.7. This implementation, however, uses a newer LU format in an Internet Draft that is work in progress. One of the changes is to expand the lifetime field to 32-bit to align with RA lifetime fields, which are also 32-bits [64]. For the purpose of aligning the fields to 32-bit width, each individual locator entry contains the 16-bit RESERVED field. The following Figure 6.8 shows the new LU message format. In addition, in order to effectively control the use of L64s in communication between the two hosts, new LU code values were defined. Table 6.1 describes the current work in progress Internet draft definitions.

For the purpose of the evaluation, as a prototype implementation, only the ILNP_LU_REPLACE_SOFT

ICMPv6 Locator Update message



Type: 156

Code: 0-7

Checksum: The 16-bit one's complement of one's complement sum of the ICMP message, starting with the ICMP type. For computing the checksum, the Checksum field is set to 0 .

Num of Locs: The number of 64-bit Locator values that are advertised in this message. This field MUST NOT BE zero.

Operation: The value in this field indicated whether this is a Locator Update Advertisement (0x01) or a Locator Update Acknowledgement (0x02).

Reserved: A field reserved for possible future use. At present, the sender MUST initialise this field to zero. Receivers should ignore this field at present. This field may be used for some protocol function in future.

Locator[i]: The 64-bit Locator values currently valid for the sending ILNPv6 node.

Preference[i]: The preferability of each Locator[i], relative to other valid Locator[i] values. The Preference numbers here are identical, both in syntax and semantics, to the Preference values for L64 records as specified by RFC6742[55].

Lifetime[i]: The maximum number of seconds that this particular Locator may be considered valid in unsigned 32-bit value. Normally, this is identical to the DNS lifetime of the corresponding L64 record, if one exists.

Figure 6.8: The ILNPv6 LU message structure currently being drafted.

code was implemented. This implementation and discussions around the existing RFC have contributed to the work in progress Internet draft amending the RFC6743 [3].

The new draft RFC updating the RFC6743 [3] is the cornerstone to enabling the mobility-multihoming duality using ILNPv6. Signalling mechanism is key in the operation of network protocols. The new set of LU codes enable not only the mobility-multihoming duality but also introduces further flexibility in LU operations. For example, add/delete codes can enable LU to be more efficient if the hosts are associated with many L64s and must update many CNs, while the replace operations can be used to make sure they are still all in sync by sending the whole set periodically.

6.2.5 Soft-handoff in multihoming

When conducting soft-handoffs in mobility scenarios, the communication must first transition to the *'new path'* before the *'old path'* is decommissioned in order to avoid loss. In the context of mobility-multihoming, when removing an interface from the ongoing communication, it must be done at the mobility-multihoming management mechanism itself and must be coordinated between the two hosts participating in communication.

This implementation of ILNPv6 includes a *'disabled-interface list'* to facilitate successful transfer of communication away from the interface before it is disabled. Specifically, a network interface could be added to such a list, which triggers the LU message to be sent to the other host in the communication session to notify it that a particular locator is to be decommissioned from the session, instructing the host to stop sending to that locator.

However, for this evaluation, the interface is not turned off completely, but rather, only listed in the disabled-interface list. This allows the evaluation to remain simpler in terms of the procedure, and eliminated the time required for lower layer initialisation beyond the beginning of the communication session.

6.2.6 Sysctl interfaces used to control ILNPv6 mobility-multihoming duality mechanism

Since there is no ILNPv6-specific socket application programming interface (API) or applications that can control the behaviour on a per-socket basis, ILNPv6 behaviours are applied host-wide for this proof-of-concept implementation. Therefore, to control the behaviours, the sysctl interface is used.

`net.ilnp6.handoff_mode` in the sysctl interface controls the host-wide handoff mode as shown below:

`net.ilnp6.handoff_mode = 0` — mobility with soft-handoff

`net.ilnp6.handoff_mode = 1` — mobility with hard-handoff

`net.ilnp6.handoff_mode = 2` — mobility-multihoming

The following evaluation is conducted with value 2, which was added for this implementation. Specifically, this enables mobility-multihoming in both directions — the source locator selection will load-balance on sending interfaces, and the destination locator selection will enable multi-homed receivers to receive on multiple interfaces. The load-balancing mechanism will attempt to evenly distribute the traffic based on bytes sent to and from the ACTIVE locators on both the source and the destination L64s. When a new interface is available and a new RA is received, an L64 entry is added, triggering LU if there is any remote NID entry present.

`net.ilnp6.disabled_interface` dictates which of the interfaces are not to be used. This is a text field parameter, which can hold multiple interfaces as a space-separated list. For example, for a host with interfaces `enp3s0f0`, `enp3s0f1`, `enp3s0f2`..., the following entry will force ILNPv6 to only use `enp3s0f1`:

```
net.ilnp6.disabled_interface = enp3s0f0 enp3s0f2
```

Upon changes to the list, ILNPv6 will change the L64 states accordingly and send LUs as needed.

6.3 Evaluation of continuous multihomed-mobility scenario

In order to demonstrate the capability of ILNPv6 in conducting multihoming and mobility transport simultaneously, multihoming and ‘host-wide equal-split’ load sharing mechanism was implemented in addition to the existing mobility mechanism. End-host load-sharing is a potential use case of multihoming as it can increase the available bandwidth of the host by using the existing distinct connectivities that are otherwise not utilised. To evaluate the multihoming mechanism, the following scenario was developed to conduct the following experiment.

6.3.1 Continuous multihomed-mobility scenario

Similar to the continuous mobility scenario in Section 4.5.1, this scenario considers a mobile host, equipped with multiple interfaces, in continuous movement. To evaluate the capability of the mobility-multihoming duality mechanism effectively, the scenario consists of four networks, one dedicated to the stationary CN (Network **dd** with prefix `2001:2:dd:dd::/64`), and three of them dedicated to the MN. As such, the MN has the following interfaces with respective networks:

Interface 1 — Network **aa** — `2001:2:aa:aa::/64`

Interface 2 — Network **bb** — `2001:2:bb:bb::/64`

Interface 3 — Network **cc** — `2001:2:cc:cc::/64`

The MN will begin with a single connectivity, gradually increase to three connectivities, then gradually reduce to a single connectivity at the different network on a different interface than the one it started with — here, that would be starting from network **aa** on interface 1 to network **cc** on interface 3 as shown below:

1. Start communication session using the interface 1 at the MN
2. The MN enables the interface 2 — this initiates load-balancing across the two interfaces
3. The MN enables the interface 3 — this initiates load-balancing across the three interfaces
4. The MN sets interface 1 in the ‘disabled.interface’ sysctl list — switching to two-interface load-balancing

5. The MN sets interface 2 in the ‘disabled_interface’ sysctl list — switching to a single-homed connectivity at network **cc**

The MN will then repeat this movement in the opposite direction, returning to the initial network, network **aa** on interface 1:

1. The MN enables the interface 2 — this initiates load-balancing across the two interfaces
2. The MN enables the interface 1 — this initiates load-balancing across the three interfaces
3. The MN sets interface 3 in the ‘disabled_interface’ sysctl list — switching to two-interface load-balancing
4. The MN sets interface 2 in the ‘disabled_interface’ sysctl list — switching to a single-homed connectivity at network **aa**

All of the handoff movements, except when all three interfaces are active performing 3-way load-sharing, were done with the same duration as the continuous mobility scenario, which is ten seconds, to highlight the agility of the handoff mechanism. During 3-way load-sharing, the MN remains stationary for twenty seconds, demonstrating the stability of 3-way load-sharing. In the initial phase of enabling interfaces, the `ip link set up` command is used, but disabling the interface is done via the `sysctl` interface to control the disabled ILNIPv6 interface list. When re-enabling the interface, the same disabled ILNIPv6 interface list is modified accordingly.

6.3.2 Experiment setup

The testbed used in the following experiments was described in Chapter 4, Section 4.4.

In this evaluation, additional delays were added to the paths between the CN and the MN. The additional delays were introduced at the R4 using the built-in `netem` in the Ubiquiti Networks EdgeOS. As discussed in Section 5.5.5, in reality, the paths will have varying RTT depending on the topology, distance, and other factors adding delays on the path. Longer RTT can cause misordering, which may impact the behaviours of applications. In a multihomed scenario, misordering may be more likely than the mobility scenario, as the host is utilising multiple interfaces, which are connected to multiple paths for much longer than during a single soft-handoff in a

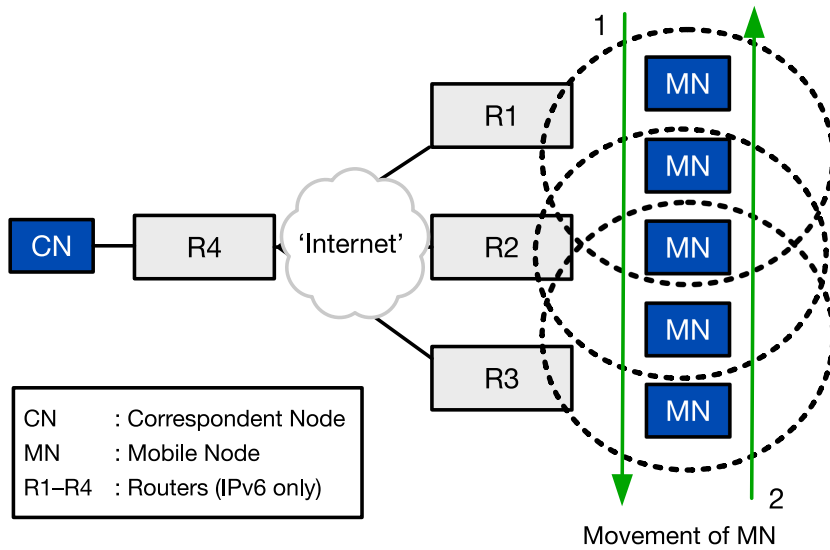


Figure 6.9: A scenario diagram describing the MN's movement for mobility-multihoming duality scenario. The arrow labelled 1 is the first movement that the MN carries out: moving from network **aa** on R1 to network **cc** on R3. The arrow labelled 2 shows the second set of movement where the MN returns from network **cc** back to network **aa**.

mobility-only scenario. During multihomed communication, different RTTs may impact how severe the misordering is, potentially impacting the behaviours of the applications. Therefore, for the multihomed-mobility scenario evaluation, additional RTTs were introduced to observe the impact of them. The different added RTTs are shown in Section 6.3.3.

6.3.3 Procedure

The following describes the procedure of the continuous multihomed mobility scenario run:

1. Enable one interface on both the CN and the MN
Specifically, the MN will enable the interface 1
2. Begin bi-directional data transfer between two hosts using iperf
3. Trigger the MN's movement scenario, which enables additional interfaces
 - (a) Enable another interface
 - (b) Wait for handoff duration

4. Repeat step 3 until all interfaces are up and enabled

At this point, the mobile node has three interfaces enabled for the first time

5. Trigger the MN's movement scenario disabling interfaces

- (a) Disable another interface

- (b) Wait for handoff duration

6. Repeat step 5 until all but one interface is up

7. Repeat steps 3–5 once such that there are two occasions where all interfaces are up, and then to return to single interface state.

8. Terminate data transfer

In this experiment, iperf2 TCP and UDP were used as payloads. In addition, the following three RTT configurations were used:

0 ms (No added delay): *LAN* equivalent

20 ms of additional RTT: *'National' delay* equivalent

200 ms of additional RTT: *'Inter-continental' delay* equivalent

Section 4.5.6 shows how these values were chosen.

6.4 Results

ILNPv6 successfully provided smooth handoffs transitioning from single-homed, to a multihomed load-shared state, and back to single-homed at the different network. A typical run from the runs in Section 6.4.1 shows individual runs for both TCP and UDP. This provides a more *'intuitive'* view of how mobility-multihoming with load-sharing functions. Section 6.4.2 shows aggregates of twenty runs per experiment configurations, highlighting how they performed overall.

6.4.1 Example runs

The following Figures 6.10 and 6.11 on pages 123–124 show example runs with no-added RTT. As shown, in both directions, the aggregate throughput remains stable, with a consistent increase in the sequence numbers observed, while the load is equally split between the interfaces active at any given time.

6.4.2 Aggregated results

The following shows the aggregated results of the mobility-multihoming scenario. They are the aggregated results of 20 runs per configurations: three delay configurations and two payloads (UDP and TCP). The flows were bi-directional, and both were used in the aggregations, labelled as CN or MN according to where the observations were made.

TCP misordering

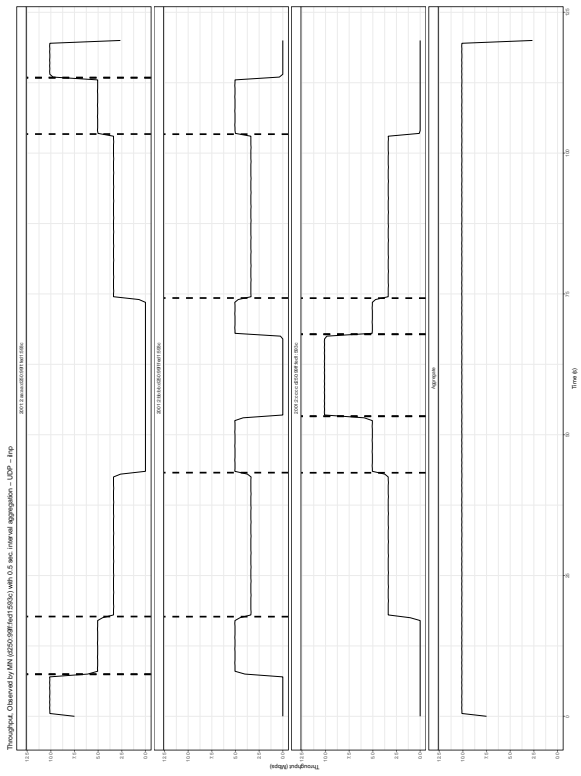
Figure 6.12 shows misorder counts for the TCP scenario. Throughout the different delay configurations, in both directions, TCP over ILNPv6 performed consistently in terms of misordering. Across the board, the misordering observed were little to none.

TCP resend

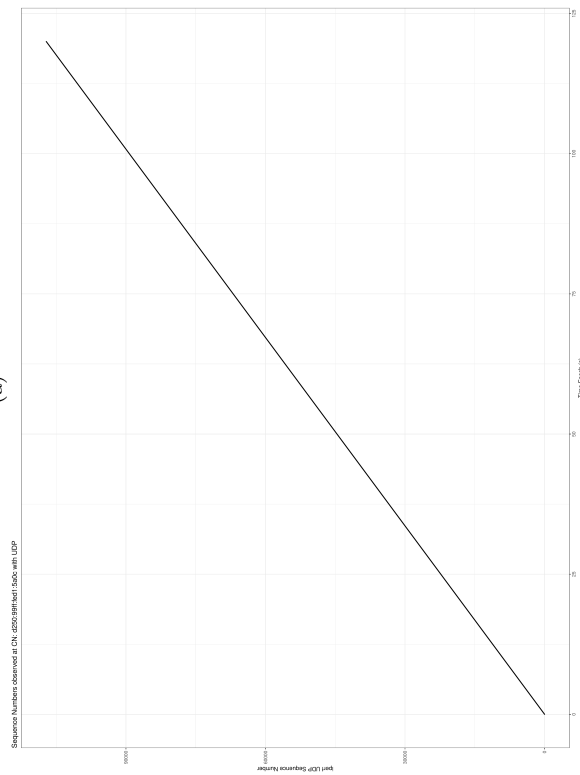
Figure 6.13 shows the TCP resend that was observed. Similar to misordering, across the different delay configurations and directions of flow, duplicate packets were negligible.

TCP throughput

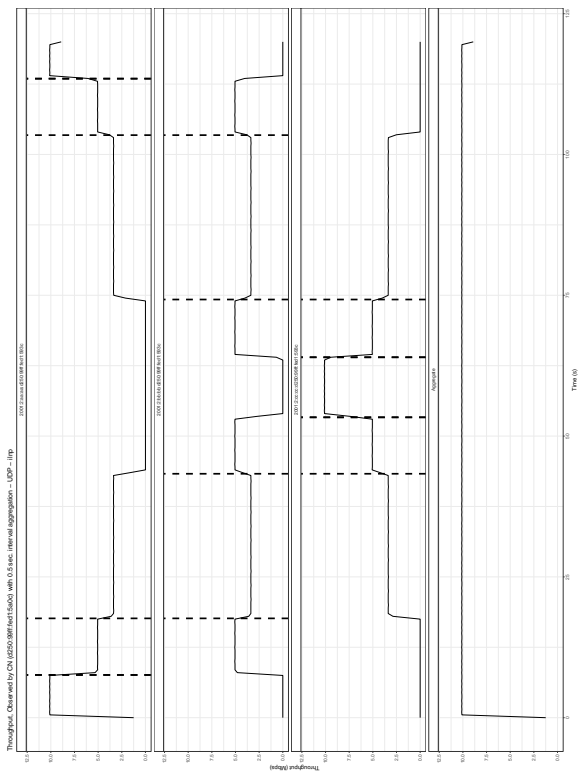
Figure 6.14 shows the throughput observed with TCP scenarios. With the target throughput of 10 Mbps, the throughput observed remained consistent. While 200 ms scenarios experienced a slightly wider distribution, the spread was, at most, 0.04 Mbps, which was a negligible 0.4% difference.



(a)



(b)



(c)



(d)

Figure 6.11: Plots showing the throughput and sequence numbers observed in typical mobility-multihoming duality iperf2 UDP scenario evaluations. Plots (a) and (c) represent the flow received at the CN. Plots (b) and (d) represent the flow received at the MN. The results showed consistent aggregate throughput at the bottom facet of plots (a) and (b), while the top three facets show throughput observed at the respective source/destination L-V, showing smooth, equal splits. The vertical dashed line shows the Locator Update messages. Plots (c) and (d) show consistent increase in the sequence numbers, indicating a consistent flow and delivery of packets.

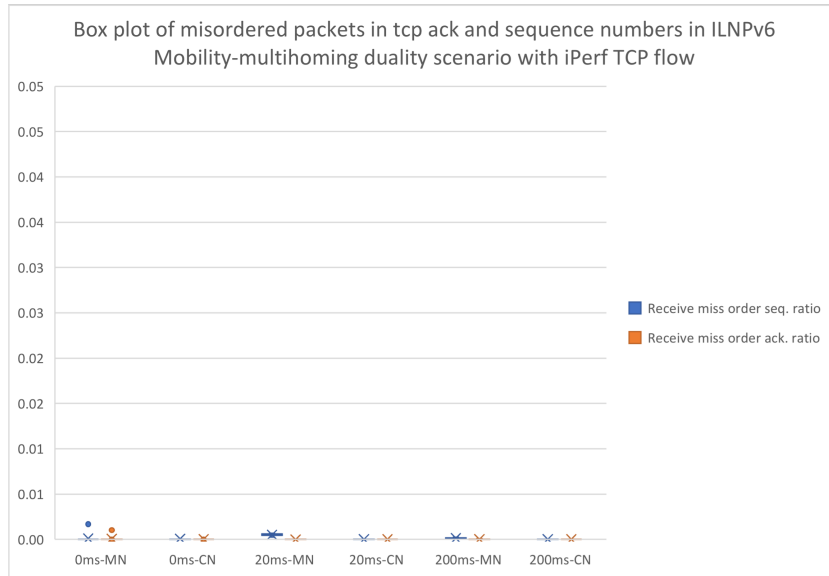


Figure 6.12: A plot showing the TCP misorder ratio for the sequence numbers and the acknowledgement numbers received. Little to no misordering was observed with both the sequence numbers and the acknowledgement numbers received. The \times indicates the mean.

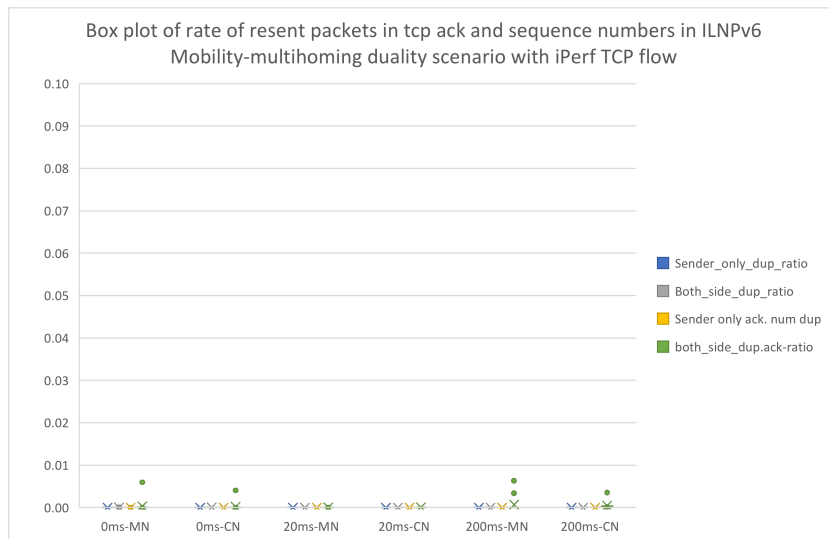


Figure 6.13: A plot showing the TCP resend ratio for the sequence numbers and acknowledgement received. No significant resend was observed for both the sequence numbers and the acknowledgement numbers. The \times indicates the mean.

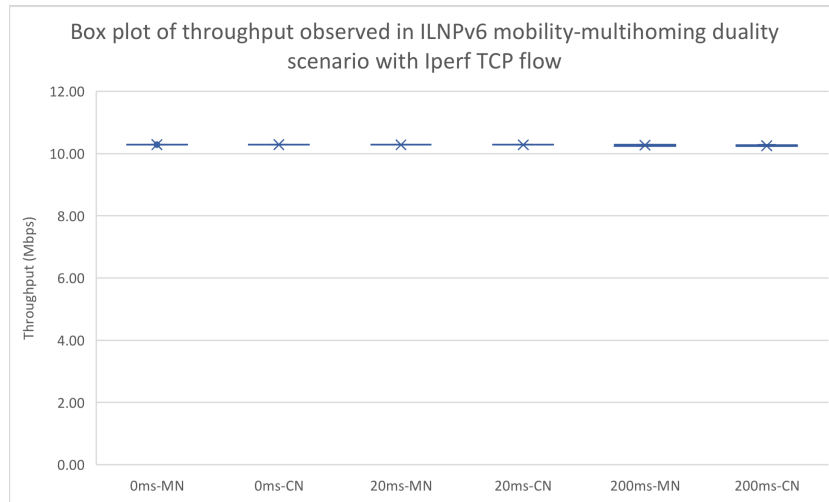


Figure 6.14: A plot showing the TCP throughput observed in the mobility-multihoming duality scenarios with the iperf TCP flow over ILNPv6. Across all delay scenarios, the throughput remained near the 10 Mbps target with negligible distribution. The \times indicates the mean.

UDP packet delivery statistics

Figure 6.15 shows the UDP packet delivery statistics. Both misordering and loss observed were less than 1% across all delay configurations and in both directions.

UDP throughput

Figure 6.16 shows throughput observed for UDP mobility-multihoming scenarios. Throughput observed remained consistent, with outliers within ± 0.02 Mbps, which was $\pm 0.2\%$ of the target throughput of 10 Mbps.

6.5 Discussion

6.5.1 Interface management in a mobile-multihomed host

As discussed, multiple physical interfaces often exist on mobile devices. The use of these interfaces may need to change to adapt to a more dynamic use of multiple interfaces. Perhaps, depending on the battery status, the host may opt to disable multihoming. Similar to how some devices step down from 5G to 4G when the demand from the user is ‘low’, not enabling

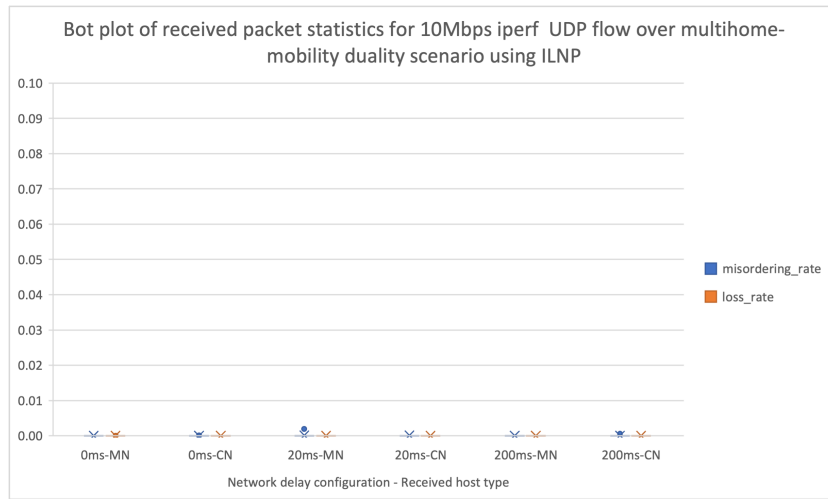


Figure 6.15: A plot showing the UDP packet statistics observed in mobility-multihoming duality scenarios with the iperf UDP flow over ILNPv6. With all scenarios, both misordering and loss rate remained little to none. The × indicates the mean.

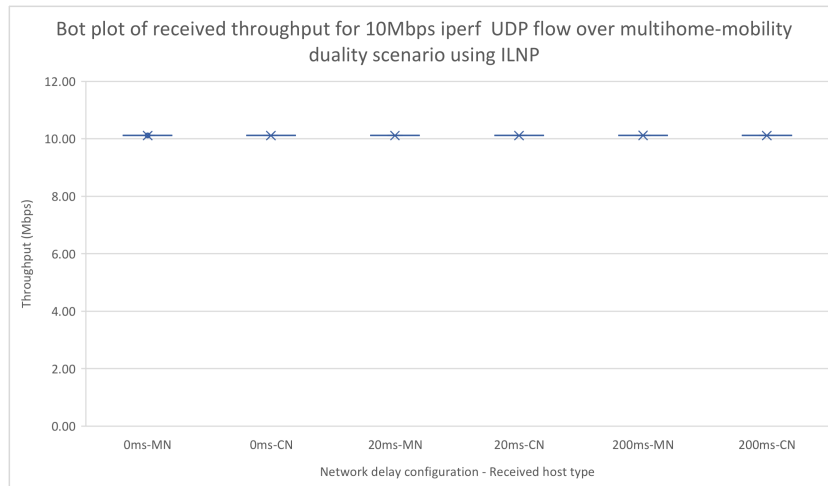


Figure 6.16: A plot showing the throughput observed in mobility-multihoming duality scenarios with the iperf UDP flow over ILNPv6. The throughput remained consistent at around the 10 Mbps target across the board with very few exceptions. The × indicates the mean.

multiple interfaces may be more optimal in some cases. Contrarily, with high demand from the user on a device that has an ample battery charge, the user could benefit from utilising all the connectivities available on the device. ILNPv6's mobility-multihoming duality mechanism will be able to adapt to such situations. In fact, with this proof-of-concept host-wide load-sharing implementation using the `sysctl` interface, it will be straightforward to enable such a mechanism. The `sysctl` interface extension described in Section 6.2.6 adds an option to remove the L64 associated with a specific interface before the physical network interface is disabled completely. With an appropriate mechanism, this `sysctl` interface will enable flexible management and usage of network interfaces at a host-wide level. Overall, these configurations and policies are dependent on the situation, application of the device, and the device itself. The results showed that the mechanism implemented in this chapter can accommodate the changes to connectivity to enable those policies required for different devices.

6.5.2 Fixed behaviour in mobility-multihoming duality load-sharing application implementation

The implementation used for this experiment is a proof-of-concept mobility and multihoming duality mechanism at a network layer as an end-host-only solution. Here, a fixed policy to enable equal-split load-sharing has been implemented as an application of mobility-multihoming duality mechanism to demonstrate the capability of the mechanism. In order to use mobility-multihoming duality in mobile applications on mobile devices in the *real world*, further work is necessary. For example, the host should be able to negotiate with the other host on how the traffic should be split, or be able to judge when a path should be decommissioned from the communication session.

The current implementation does not negotiate preferences of L64s for splitting traffic between different paths as this implementation has the fixed-equal-split policy implemented. Some connectivities may be metered, so, a user may prefer not to use them for certain traffic, or perhaps a specific path may have different capacities, costs, or preferable (or non-preferable) delay profiles. Instead of the fixed behaviour of using all of the available interfaces except for those listed not to, in an equal-split fashion, the user should be able to set a policy dictating how

the different interfaces and their paths are used. In addition, the use of interfaces may also be application specific. For example, a video streaming application may always want to opt for Wireless-Fidelity (Wi-Fi) and avoid cellular connectivity for transmitting the video itself but prefer to utilise any connection available for the signalling traffic. Moreover, some of the decision making should be automatic in order for it to be deployed widely. Such mechanisms would need an additional control-plane or management-plane interaction, which could be application specific. This implementation does not constrain such interactions, and, indeed, as with the discussions in Section 6.5.1, this enables such mechanisms to be designed and implemented in the future. This can be explored in future work as mobility-multihoming duality is now possible with this fixed behaviour.

Similarly, knowing when the host needs to decommission an interface requires other lower layer technology specific information:

1. Lower layer signalling
2. Signal strength/quality
 - (a) received signal strength indicator (RSSI)
 - (b) signal-to-noise-ratio (SNR)
3. Underlying radio modulation technologies

Currently, IPv6 RA is the only signal used to trigger changes to connectivity but this can easily be extended.

The previous work and this work have both shown that ILNPv6 LU is a simple and fast signalling mechanism, allowing agile movement, now with host mobility-multihoming duality. Exactly how it can be best utilised needs to be investigated in future work. The current mobility-multihoming duality has the fixed behaviour for host-wide load-sharing as an application of mobility-multihoming duality. In addition, as this is a proof-of-concept implementation, there are some assumptions regarding the starting state. One assumption here is that the host only has one interface available at the beginning of the communication, i.e. it has not considered cases where, at the beginning of communication, multiple interfaces are available to be used. The scenario only considers cases where both sides only use one interface at the beginning, allowing

the new RA to trigger signalling to add new L64s to the ILCC entries. This is done to simplify the application to carry out the evaluation of mobility-multihoming duality mechanism.

Another aspect to consider is the different behaviours that different layer 2 technologies exhibit. Those different layer 2 technology behaviours may impact how different interfaces should be used for different applications.

6.5.3 Boot-strapping multihoming transport

One of potential scenarios a multihomed host may encounter is a scenario where the interfaces available to it are already enabled at the OS level, that is, they all have the relevant IPv6 prefix from the router via RA and are ready to send/receive packets. In such a scenario, it is unclear how the current implementation will handle the multiple interfaces. Therefore, as described at the end of Section 6.5.2, the experiment scenario specifically begins with a single connectivity at the MN, before adding another.

As described earlier, the way the mobility-multihoming duality mechanism can be used varies depending on the application and the environment it is being used in. However, in the context of deploying host-wide load-sharing mechanisms with ILNPv6, the following approach may be an option to manage multihomed transport with load-sharing:

In the situations where multiple interfaces are in active use at the start of a new communication session, a ‘bootstrap’ procedure is needed to be built into the load-sharing mechanism implementation. Following is an example of such procedure:

1. When multiple interfaces are available, pick one and begin a communication session with the CN, i.e. a single initial pair of Identifier-Locator Vector (I-LV) resolved via any mechanism the hosts used originally.
2. Once the communication session is established, send an LU with all of the L64s from available interfaces.
3. Optionally, additional interfaces L64s could be set to `VALID` — preventing transmission but allowing receiving — until the relevant LU-Ack is received.

The above should be done for each new NIDs to correctly handle each CNs to make sure that the other side holds the relevant locator sets.

Such a mechanism is not uncommon — other multipath transport mechanisms such as MP-TCP, Mutlipath extensions for QUIC, and Stream Control Transmission Protocol (SCTP) use a ‘bootstrap’ phase with the initial single connection to initialise the rest of connectivities available.

Again, the limitation described here is purely for the sake of simplifying the evaluation, and for simplicity in building an application to evaluate the control plane developed in ILNPv6 to demonstrate the contribution of the mechanism.

6.6 Summary

This chapter presented the novel mobility-multihoming duality mechanism using ILNPv6. The ILNPv6 implementation here enabled true and completely agile multiple connectivity for typical end-hosts with standard IPv6 connectivity, using existing open-source software, and with off-the-shelf commercial networking devices. The results showed little or no impact on the overall throughput and loss, providing a smooth transition from single-homed connectivity to multi-homed connectivity, while being able to completely depart from the initial point of attachment and return to it without any issues. This enables *true dynamic mobility and multihoming duality* communication through an end-host solution, without requiring changes to the network or the application.

Chapter 7

Application level impact on real-time video

The behaviour and features of the underlying network protocol will have an impact on the behaviour of applications utilising it, thus, it has the potential to influence users' experience — Quality of Experience (QoE). Therefore, when evaluating the effectiveness of the features implemented in a particular network protocol, investigating the impact on the user experience is important. One of the common modern applications of the Internet is real-time video streaming. Whether it is a live stream or a video call, more and more devices on the Internet are capable of streaming real-time videos. This chapter presents the impact of both continuous mobility performance and multihoming-mobility duality handoff performance on real-time videos using objective user experience metrics.

7.1 Rationale for QoE analysis of real-time video application over ILNPv6

The previous work evaluating Identifier Locator Network Protocol v6 (ILNPv6) was done at the protocol level performance using benchmark/diagnostics tools such as iperf and Internet Control Message Protocol v6 (ICMPv6) echo, and no QoE measurements have been taken yet.

Real-time video transmission is an appropriate candidate for evaluating the application-level impact of the network protocol performance. First, it is a challenging scenario for network mobility and multihoming. To achieve a short latency, the buffer is often kept short, leaving little to no room for error such as loss or misordering. Any loss or delayed packets beyond the size of the video buffer directly impacts the visual quality of the image, impacting the user-experience (UX). Second, as described in Section 2.13, the use of video applications over the Internet is becoming more popular, and with most modern mobile devices equipped with cameras, real-time videos on those devices are also common with live-streaming and video calls.

To evaluate the utility of ILNPv6, it is crucial to examine the effectiveness of its functionalities using *'real'* applications with *'real'* traffic examining objective QoE metrics. The following experiment examines the potential impacts to the QoE for the user by analysing both packet level metrics as well as objective QoE metrics in a repeatable, reproducible manner using Real-time Transport Protocol (RTP)/User Datagram Protocol (UDP) for applications sending and receiving real-time video payloads.

7.2 ILNPv6 backwards compatibility and legacy application support

ILNPv6 is able to support legacy applications without modifying them. The applications can benefit from ILNPv6 as long as they use names to refer to a host. The ability for legacy applications to operate with ILNPv6 and its advanced features such as mobility and multihoming provides a way to not only evaluate how applications would behave with such unique network protocol features, but also to gain some insight into how ILNPv6 would perform with the many Internet Protocol (IP)v6-capable applications that exist today.

7.3 Impacts of network layer performance on application layer performance

One of the ways of categorising networked applications is by their time-sensitiveness: time-sensitive ‘*real-time*’ applications and non-time-sensitive applications. For the latter, often, the network layer performance, such as loss and latency, does not hugely impact the end result with great sensitivity. One example may be a simple Hyper Text Transfer Protocol (HTTP)/Hyper Text Transfer Protocol Secure (HTTPS) request, where if there is loss, underlying transport protocols, such as Transmission Control Protocol (TCP), will recover it by resending the lost payloads; and high latency may contribute to slow load time or responsiveness, but does not cause a grave concern per se. The user will still receive content without any impact on the data. However, more time-sensitive applications, such as Voice over IP (VoIP), live video stream, and video conferencing, can be impacted by the network layer performance much more severely. Specifically, for live video streaming, a jitter in latency or packet loss can cause issues such as skipping frames or noise artefacts appearing on frames. To evaluate the impact of the handoff, a real-time application is an appropriate candidate as it is sensitive to network level performance changes such as jitter and loss.

7.4 ‘Real-time’ video transport experiment setup

In the context of evaluating the performance impact on ‘real-time’ video applications, pre-recorded clips were used as payloads, streamed from the Correspondent Node (CN) to the Mobile Node (MN). While it is true that those applications may have the video captured and encoded on-the-fly, then transmitted to the other host, pre-recorded clips allowed repeatable experiments to be conducted without potential uncertainty in encoding overhead. As a proxy to real-time video, pre-recorded clips were transmitted as if they were generated/captured and streamed live, by using appropriate transport layer protocols — RTP/UDP — and applications utilising them — `ffmpeg` and `vlc`.

As payloads, two clips from Tears of Steel (ToS) and Big Buck Bunny (BBB) were used, at three different resolutions. This is summarised in detail in Table 7.1 on page 136.

7.4.1 Video payload selection and preparation

In order to evaluate the effectiveness of the seamless handoff and its impact to real-time video streaming, two high definition (HD) creative commons license clips were chosen. The video payloads chosen for the transmission were pre-encoded, trimmed clips from: Tears of Steel (ToS), a short live-action movie with 3D effects; and Big Buck Bunny (BBB), a 3D animation short film.

In order to perform continuous movement with multiple handoffs, the duration of evaluation was set to 120 seconds, thus the videos chosen were trimmed to that length accordingly. The choice of the clips follows a similar approach to [72]. To avoid static black screen with little changes in the frames, for Tears of Steel, 120s of the clip was trimmed from the 6:00.00 mark where there were various actions in the scene. As Big Buck Bunny immediately starts without a static image, 120s of the clip was trimmed from the beginning of the video. The experiment was carried out with the following resolution formats: 720p (HD), 1080p (FHD) and 2160p (4K UHD). A source uniform resource locator (URL) of each clip and its relative resolutions are summarised in Table 7.1.

While the scope of the experiment was to evaluate ‘real-time video streaming’, to enable a repeatable evaluation, the pre-recorded clips were streamed to emulate real-time clips using the Real-time Transport Protocol (RTP)/UDP. MPEG Transport Stream (MPEG-TS) was chosen as the container along with H.264 encoding, as they are a commonly used container and a commonly used encoding format in real-time video streaming applications respectively. The common real-time video application Microsoft Teams uses H.264 codec in their video conferencing feature as shown in Figure 7.1 in the Microsoft Teams Call Health pane. The source video files were downloaded in each resolution from the respective URLs shown in Table 7.1. The clips were trimmed and re-packaged into MPEG-TS (.ts) containers using ffmpeg¹ software. Those MPEG-TS files were then stored at the CN.

¹See Appendix E for further detail for reproducibility

Footage	Big Buck Bunny	Tears of Steel
Codec	H.264	
License	Creative Commons Attribution 3.0	
Original duration	10:34	12:34
Duration used	120 seconds	
Original framerate	30 fps	24 fps
Framerate used	24 fps	
Footage type	3D animation	Live action with 3D effects
<hr/>		
720p		
Dimension	1280 × 720 px	1280 × 534 px
Original URL	https://download.blender.org/peach/bigbuckbunny_movies/big_buck_bunny_720p_h264.mov	http://ftp.nluug.nl/pub/graphics/blender/demo/movies/ToS/tears_of_steel_720p.mov
<hr/>		
1080p		
Dimension	1920 × 1080 px	1920 × 800 px
Original URL	http://distribution.bbb3d.renderfarming.net/video/mp4/bbb_sunflower_1080p_30fps_normal.mp4	http://ftp.nluug.nl/pub/graphics/blender/demo/movies/ToS/tears_of_steel_1080p.mov
<hr/>		
2160p		
Dimension	3840 × 2160 px	3840 × 1714 px
Original URL	http://distribution.bbb3d.renderfarming.net/video/mp4/bbb_sunflower_2160p_30fps_normal.mp4	http://ftp.nluug.nl/pub/graphics/blender/demo/movies/ToS/tearsofsteel_4k.mov

Table 7.1: A table showing the properties of each footage. The average throughput was derived from 30 runs.

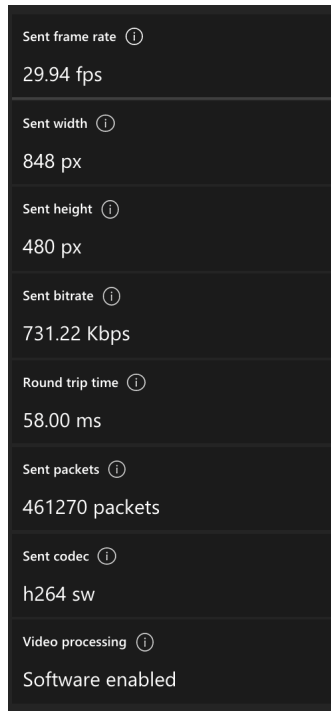


Figure 7.1: A screenshot from Microsoft Teams during a call in the ‘call health’ pane, indicating that H.264 software encoding is used for the video.

7.4.2 Added delays to round trip time (RTT)

On the Internet, hosts experience various lengths of round trip time (RTT) to the correspondent host. This depends on the topology, the physical distance, and physical technologies within the path. In a time sensitive application, this can potentially impact the way that the application behaves. In this experiment, same as Section 6, the following delay configurations were used:

0 ms (No added RTT): *LAN* equivalent

20 ms added RTT: *‘National’ RTT* equivalent

200 ms added RTT: *‘Inter continental’ RTT* equivalent

Section 4.5.6 shows how these values were chosen.

Metric	Definition	Comment
n_{finite}	Numbers of runs containing, non-Inf., finite PSNR value in one or more of the frame.	Indicates how many runs contained finite, <i>imperfect</i> frames and used for r_{inf} and other PSNR finite value calculations.
n	Total number of complete runs evaluated in this study.	Complete run is a run with expected number of frames received.
$\frac{n_{\text{finite}}}{n}$	Ratio of runs containing finite PSNR values to complete runs evaluated in the study.	Indicates ratio of <i>perfect</i> runs in terms of per-frame PSNR analysis.
r_{inf}	Mean ratio of Inf. PSNR frames to finite PSNR value frames.	Indicates on average how much of the frames were disturbed.
PSNR_{min}	Mean of minimum PSNR value observed	Indicates how severely on average the clip was impacted in terms of observation with PSNR values.
PSNR_{Δ}	Difference between minimum and maximum mean PSNR observed.	Indicates on average, how much did PSNR fluctuate within the runs with finite values observed.
Mean SSIM	$\frac{\text{SSIM}}{\text{No. Frames}}$	Average SSIM
Median SSIM	Second quartile, median of SSIM observed across the clip.	
Delivery rate	$\frac{\#\text{RTP seq. num. observed}}{\#\text{RTP seq. num. expected}}$	Describes at network protocol level, the ratio of packets delivered.

Table 7.2: A table showing metrics used with comments illustrating the purpose of each metric.

7.4.3 Metrics

In order to evaluate the network layer performance and its impact on the RTP application performance, both packet level metrics and frame-by-frame image analysis metrics were used to evaluate the QoE objectively. At the network level, RTP sequence numbers were used to evaluate the packet delivery metrics — misordering and loss. At the application level, structural similarity index (SSIM) and peak signal-to-noise ratio (PSNR) were measured frame-by-frame, then aggregated to evaluate the impact throughout the entire flow as the handoffs occurred continuously. Table 7.2 summarises different metrics derived from those measurements.

Structural Similarity Index — SSIM structural similarity index (SSIM) is an objective QoE metric comparing an image to its original to measure the visual difference. Specifically,

SSIM is concerned with the topology of the image. An SSIM value of 1 is considered identical, whereas, as it approaches 0, the image is considered *different* from the original. As it approaches -1, the image is ‘inverted’ topologically.

In the context of videos, since a video is a series of images, frame-by-frame analysis will show the quality of the image per-frame, which can then be aggregated to provide an overview of the entire clip. Since SSIM always gives a finite value, a numerical comparison is straightforward. Here, the mean SSIM of the clips recorded at the MN were taken and the distributions across the thirty runs were observed.

How different SSIM values represent image qualities SSIM is a measurement of image quality, whereby the metric focuses on the topology of the values to evaluate how similar the two images are structurally as the name suggests. Unlike metric like PSNR, SSIM have the upper, and lower boundaries, making numeric analysis much more straight forward. SSIM indicates how similar the two images are, with upper bound of 1.0 While the original paper on SSIM [73] indicates it can be negative, does not indicate the specific implication of such values within the paper. However, Wang has also authored a chapter titled “Structural Approaches to Image Quality assessment” [4] in a book, “Handbook of Image and Video Processing, 2nd edition”, where he further illustrates the impact of negative SSIM. In this chapter, Wang described that “...negative structural similarity values correspond to the cases that the local image structures are inverted.”. The figure 13 in the chapter (which is shown below in Figure7.2) illustrates what negative SSIM may look like, which was produced by having inverted local image structure.

Peak Signal-to-Noise ratio — PSNR peak signal-to-noise ratio (PSNR) is another objective QoE metric to observe image quality. it is, in essence, a ratio of maximum possible power to mean square error on a logarithmic scale. Due to the nature of the function, as the error reduces, the resulting value approaches infinity. Hence, when there is no visual fidelity loss, PSNR output results in *infinite*. Unlike a video encoding application, where the impact of the encoding algorithm is more uniform across the entire footage, the nature of disruption to the image quality was limited to the duration of handoffs in this evaluation. Therefore, when there were no handoffs occurring, we would observe *infinite* values. Similar to SSIM, the quality degradation

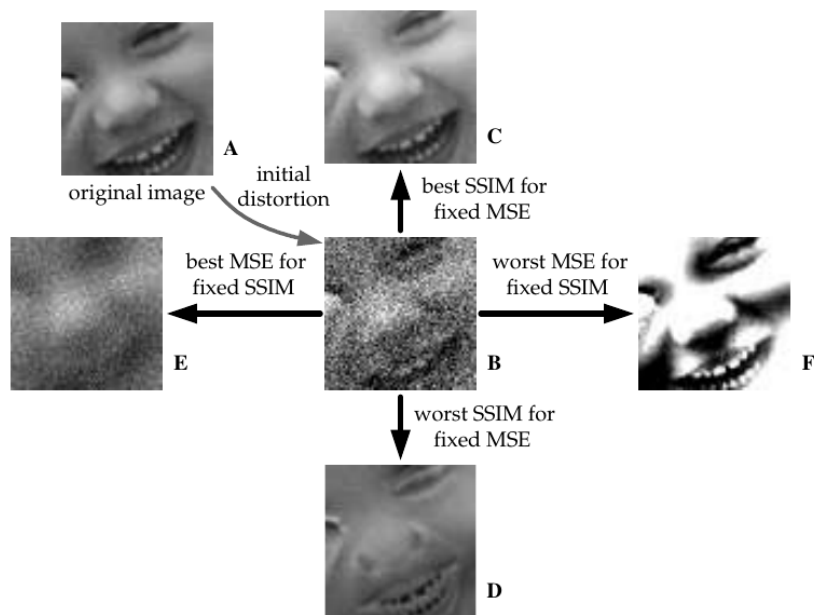


FIGURE 13 Demonstration of image stimulus synthesis for performance comparison of mean squared error and structural similarity index.

Figure 7.2: A figure describing relationships between MSE and SSIM with regards to different distortions applied to an image in [4]. Example D in particular has a negative SSIM; in this image, the local image structure is inverted.

is measured by comparing the images to the ‘originals’; in this context, the received clips that were captured at the receiver of the stream were compared against the clips that were streamed.

Since the values of PSNR are not always finite, in order to evaluate the impact on PSNR with the intermittent disruptions of quality, taking the mean or the median of the value across the entire run was not feasible. Therefore, taking a count of *finite* PSNR frames and separating them from *infinite* values was necessary. To describe the impact, the following tuple was used:

$$\langle r_{\text{inf}}, \text{PSNR}_{\text{min}}, \text{PSNR}_{\Delta}, \frac{n_{\text{finite}}}{n} \rangle \quad (7.1)$$

where:

n_{finite} = No. of runs that contained a frame with finite PSNR values (i.e. not Inf.).

n = No. of total runs

r_{inf} = Mean ratio of frames with infinite PSNR value

PSNR_{min} = Mean of minimum PSNR observed (dB)

PSNR_{Δ} = Difference of mean maximum PSNR and mean minimum PSNR observed (dB)

The n_{finite} describes how many of the runs were ‘*imperfect*’ where not all frames yielded to infinite value. r_{inf} describes the mean ratio of frames with infinite value to frames with finite value. This describes, on average, what percentage of the frames were ‘*perfect*’. PSNR_{min} describes the mean minimum PSNR value that was observed in those *non infinite value frames* — i.e. *finite value frames*, observing the worst case scenario. The mean maximum finite PSNR value was also taken. Finally, PSNR_{Δ} describes the difference between the mean PSNR_{min} and mean PSNR_{max} . A larger PSNR_{Δ} suggests, on average, there was a larger variance in quality difference.

The following tuple describes the ideal scenario:

$$\langle 1.00, \text{N/A}, \text{N/A}, \frac{0}{30} \rangle \quad (7.2)$$

The first element confirms that the mean value of ratio of infinite frames to finite frames was 1.00 — i.e. on average all frames yielded infinite PSNR values. As none of the runs included finite values, the minimum was not calculated. Similarly, no maximum PSNR was calculated, thus no minimum–maximum Δ was calculated. The last element describes that all runs and its frames yielded PSNR value of infinite, i.e all runs were *perfect* in the context of PSNR measurement.

7.4.4 Data collection and analysis

To analyse the quality of the live video stream and the impact of network protocol performance, both the layer 3 packets and the received video streams were captured. To record the video streams, they were ‘dumped’ directly on to the file system using the receiver software, `v1c`.

This reduced the possibility of the recorded files being affected by the receiver software re-interpreting the clips before they were recorded into a file. This method of data collection reduced performance impacts on the end-host and reduced the potential impact on the recorded clips due to potential performance issues. Specifically, as the resolution increase, decoding of the clips and re-encoding before saving them would increase the load on the end-host, which may impact the overall performance of the system, and thus, impact the results collected. Such system-wide performance impact could affect the other data collections such as the packet capture. The packet capture was conducted using the `tcpdump` program with `libpcap`, used in the previous chapters. In order to limit the storage usage and the CPU load, the `snap length (-s)` option was set to 128 bytes. It limited redundant data collection of the actual clip in the payload portion of the packets, while still collecting RTP header information in the payload. The packet capture allowed analysis of time stamps, payload lengths, and the RTP sequence numbers of the RTP packets, providing insight into how ILNPv6 and Mobile IPv6 (MIPv6) performed in continuous handoff scenarios at the network level.

In order to analyse the received clips, the Video Quality Measurement Tool (VQMT) [74] program, ported for Linux distribution systems [75], was used to process the recorded clips frame-by-frame. To feed a clip into the VQMT, the received clip in MPEG-TS H.264 format was encoded to YUV format using `ffmpeg`. Once the clip is converted, the VQMT tool was used to generate SSIM and PSNR of each frame. The analysis and aggregation of the generated frame-by-frame video quality metrics were done using R scripts. The analysis of the RTP packets were similar to that of the `iperf2` UDP packets, using the same script with small changes to allow the RTP packet sequence numbers to be parsed.

7.4.5 Testbed

In the following real-time video evaluations, the following testbed, as presented previously in Chapter 4 Section 4.4, was used to conduct both continuous-mobility and mobility-multihoming scenarios.

As shown in Figure 7.3, the CN has access to R4, which is connected to the rest of the routers and network devices. The MN has access to R1–R3, three routers connected to R4. The delays

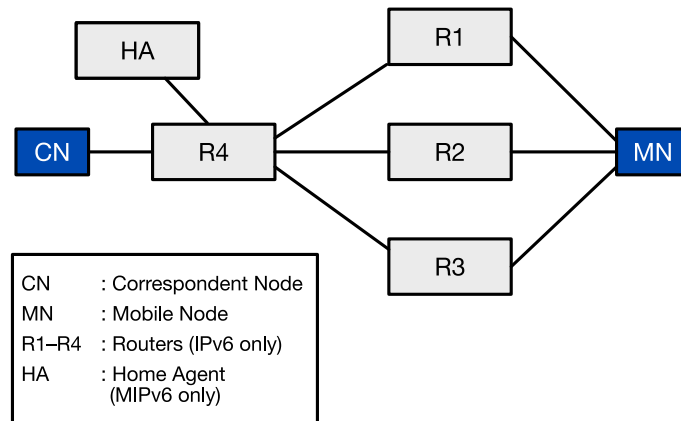


Figure 7.3: The physical topology diagram of the ILNPv6 testbed.

described in Section 7.4.2 are applied on R4 in Figure 7.3, allowing a single point of control for all connectivity between the CN and the MN.

In order to send the MPEG-TS clips across the network, `ffmpeg` is used at the CN. At the MN, `vlc` was used to receive and save the clips. Neither of the applications were modified in any way to function with ILNPv6. They were obtained from the public Debian package repository as standard binaries for Debian 9 stretch.

7.4.6 Reference point

To observe the baseline behaviour, a single run result was collected without any movement on a single, fixed, ethernet connection without any handoffs. Figure 7.4 on page 144 shows both Tears of Steel and Big Buck Bunny throughput at 1080p without any added RTT on a fixed Ethernet connectivity. This was done using IPv6, without any mobility or multihoming mechanism in place to provide a reference point as to how throughput fluctuates since the clips were encoded in variable bit rate (VBR).

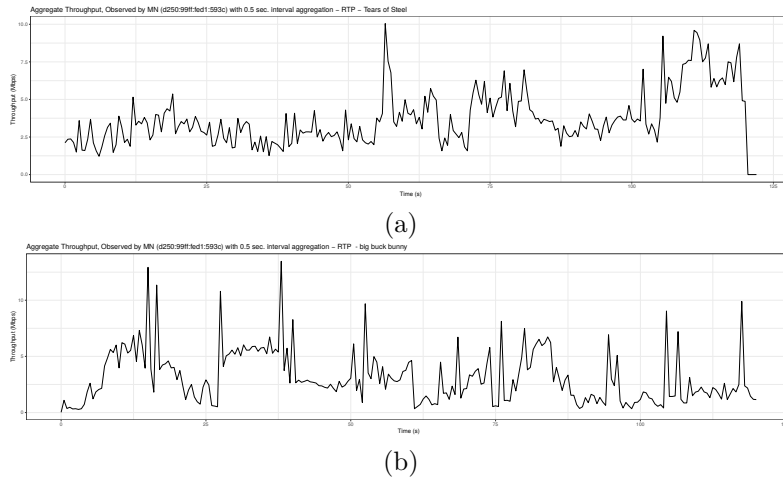


Figure 7.4: Plots showing reference throughput with a fixed connection. (a) shows the throughput plot of the 1080p Tears of Steel clip without any added RTT.(b) shows the throughput plot of the 1080p Big Buck Bunny clip without any added RTT.The width of the plots is matched approximately with the following example run plots.

7.5 ‘Real-time’ video transport evaluation with continuous mobility

As discussed, time-sensitive applications are sensitive to the stability of the underlying data transmission. Chapter 5 showed that ILNPv6 soft-handoff mechanism provided a stable, consistent, and smooth handoff compared to MIPv6. This evaluation investigates how the difference in the network layer handoff performance impacts the resulting application layer performance using objective QoE measurements for a continuous mobility scenario.

7.5.1 Scenario

Similar to the previous continuous mobility scenario, the mobile node will continuously move through different networks, while receiving and capturing the real-time video payloads from the stationary CN. The following scenario was used in this particular real-time video transport with continuous mobility:

1. Enable one interface on both the stationary CN and the MN
2. Begin RTP video transfer on the CN, begin the video capture on the MN

3. Trigger a handoff:
 - (a) Enable an additional interface
 - (b) Wait for a set handoff-duration
 - (c) Disable the previous interface
 - (d) Wait for a set duration between handoffs
4. Repeat steps in 3 until the end of video transmission
5. Terminate applications performing RTP video transfer and collection

Section 4.5.5 illustrates how the overlap duration of ten seconds was used. Following Section 4.5.6 shows how RTT configurations were chosen.

7.5.2 Results

The following shows the results from the RTP continuous mobility evaluation.

Typical Runs

Figures 7.5 and 7.6 show plots from the typical individual runs. The SSIM is plotted frame-by-frame; throughput and sequence numbers of RTP packets are plotted over the duration of the data transfer. The plots highlight the impacts of underlying data transfer on the resulting visual quality.

While the frames and the packet capture duration do not correspond 1:1, with MIPv6, the occurrence of the large SSIM dips matches with the gaps in sequence numbers. The dips that MIPv6 experienced went below 0 — the negative SSIM indicated that the topology of the image had ‘inverted’, see Appendix E, Section 7.4.3 for further detail. Meanwhile, ILNPv6 displayed no gap in the observed RTP sequence numbers and the SSIM remained at 1.00 throughout the clip.

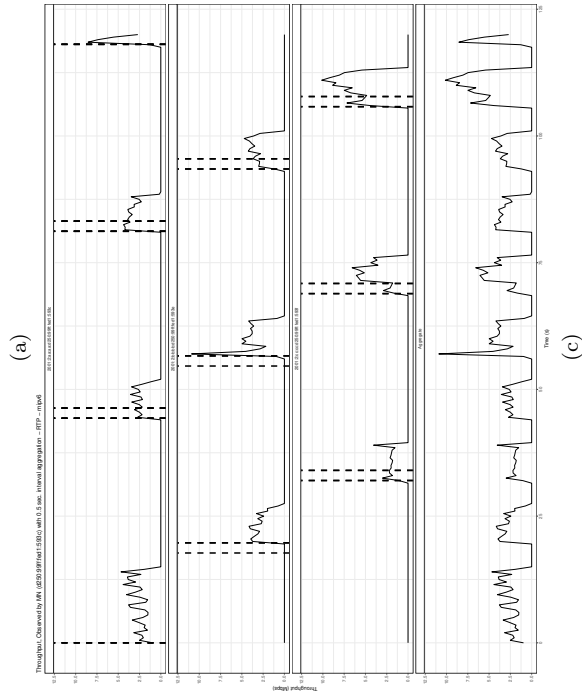
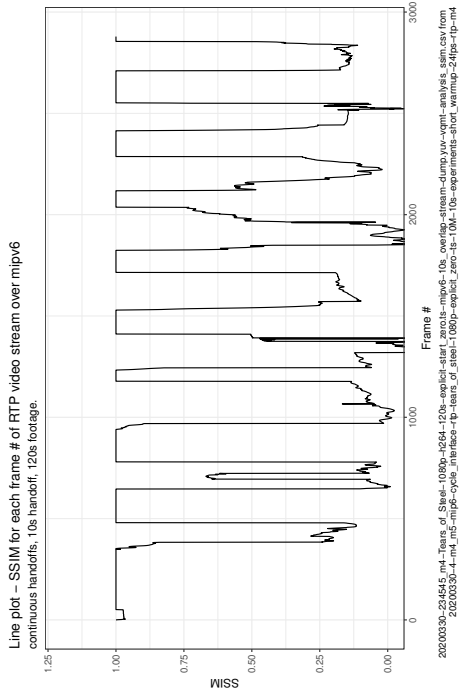
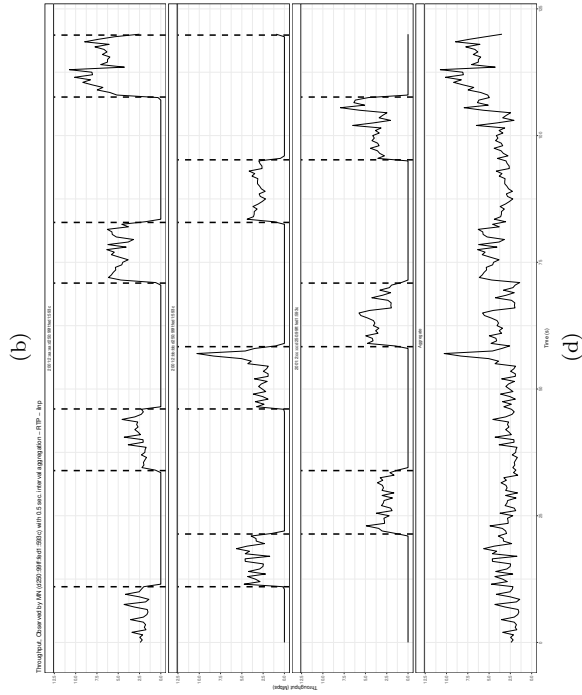
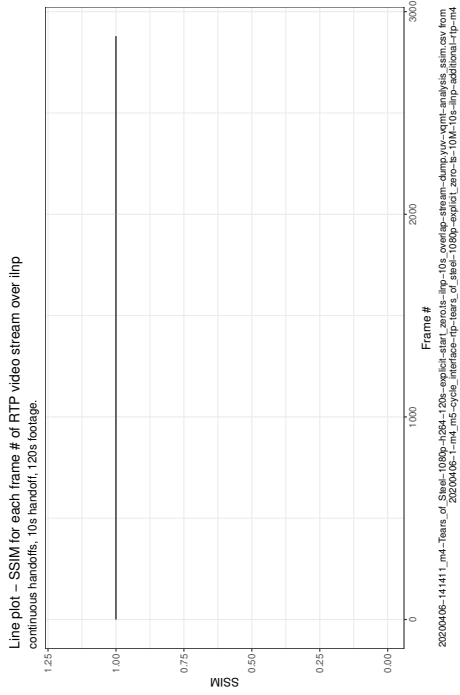
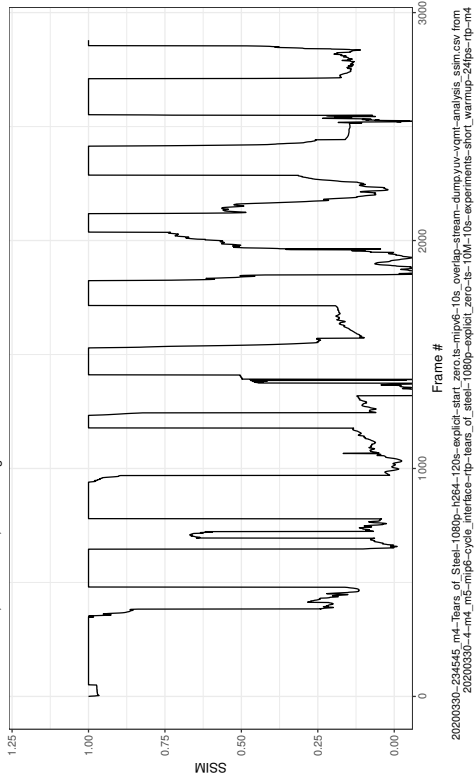
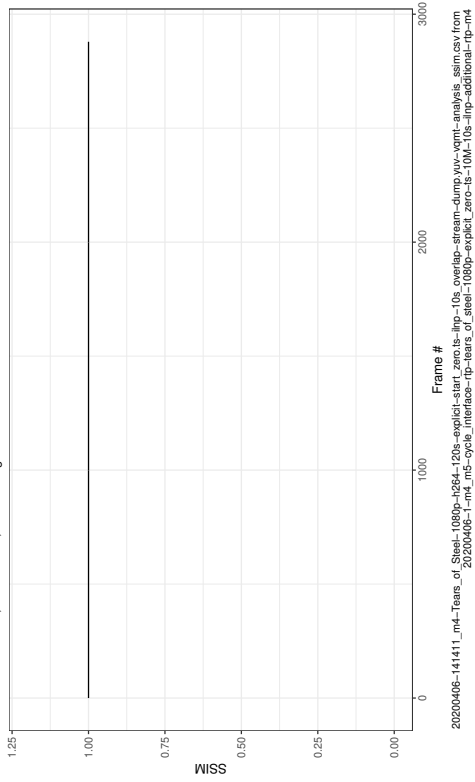


Figure 7.5: The two plots on the left are from a ‘typical run’ with MIPv6. The top two graphs represent the SSIM observed in the individual frames from the 0th to 2879th frames (120s at 24 fps, totalling 2880 frames). The bottom two graphs show the throughput observed during the transmission of the videos over respective network protocols to individual network interfaces, as well as the aggregate of the throughput. The throughput on the individual networks fluctuates as the node moves from one to another. The bottom graph labelled aggregate is the total throughput of all three networks at a given time on the x-axis. While the bottom graphs and the top graphs have different units on the x-axis, they are correlated — the top graphs are the resultant quality of the video transmitted at the rate shown in the bottom graphs. SSIM is a common metric for video quality, where 1.0 indicates that the footage is identical to its original footage with no quality degradation. Anything lower than 1.0 indicates loss of quality/mismatched image. The video was encoded as variable bit rate, which resulted in the fluctuations in aggregate throughput. As the same footage was transmitted using the same application, only with different network layer protocols, the aggregate throughput should look the same. However, in the MIPv6 results, the aggregate throughput seemed to have significant intervals of duration with 0 Mbps throughput, where no data was received by any interfaces. While on ILNPv6, there were no such gaps in data transmission. The ILNPv6 SSIM plot shows no changes over time; unlike the ILNPv6 result, the MIPv6 SSIM plot shows large changes of SSIM away from 1.0 for the same number of occurrences as the throughput gaps in the aggregate plot. Since there were no large spikes of throughput to recover the data not received in those durations, there was likely a loss. In fact, the next page shows sequence numbers on RTP packets received across time series supporting this observation.

Line plot – SSIM for each frame # of RTP video stream over mipv6
continuous handoffs, 10s handoff, 120s footage.



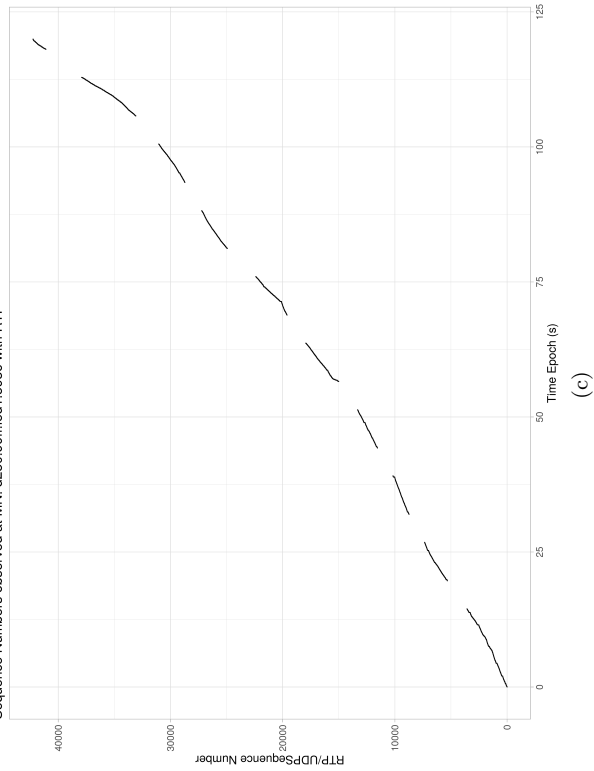
Line plot – SSIM for each frame # of RTP video stream over linp
continuous handoffs, 10s handoff, 120s footage.



(a)

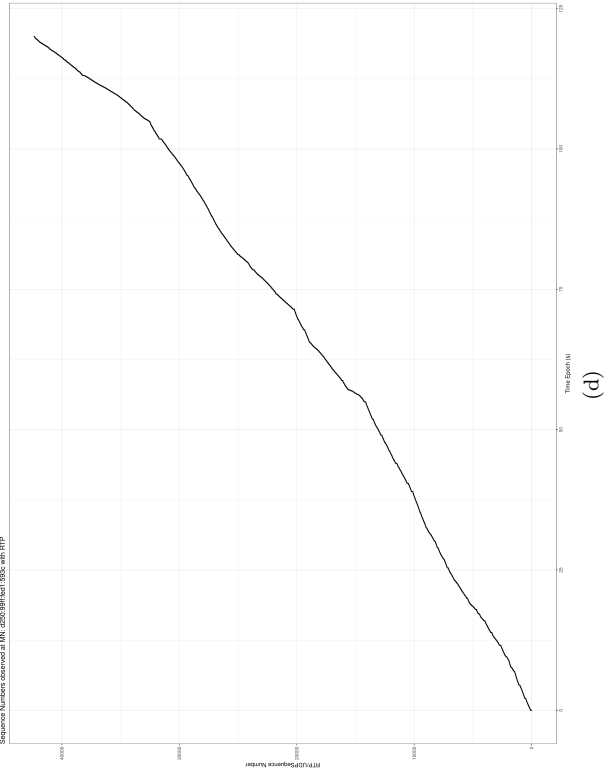
(b)

Sequence Numbers observed at MN: 4250:99ffied1:593c with RTP



(c)

Sequence Numbers observed at MN: 4202:99ffied1:893c with RTP



(d)

Figure 7.6: The two plots on the left are from a typical run with MIPv6. The two plots on the right are from a typical run with ILNP. They are both from the 1080p Tears of Steel scenarios identical to the runs in the previous set of graphs. The top two graphs represent the SSIM observed in the individual frames from the 0th to 2879th frames (120s at 24 fps, totalling 2880 frames). The bottom two graphs are RTP sequence number graphs during the transmission of the video over respective network protocols. As RTP sequence numbers start from a random number, for the purpose of plotting, the first sequence number was subtracted from the sequence numbers. Similar to the previous set of plots, the top graphs show the resultant quality of video received with the sequence numbers shown on the bottom graphs with respective timing. This set of graphs are also from the same set of runs in the previous page. On the bottom-left graph, it is evident that as the time passed, some of the sequence numbers were missing, which suggests those specific packets carrying the video were lost. Meanwhile, the ILNPv6 result on the right does not have any gaps, which suggests that there was no loss. This is in line with the previous result with 0 throughput durations on the MIPv6 run.

Example visual noise artifacts

Figures 7.7 (a)–(f) show the side-by-side comparison of visual noise artifacts observed with MIPv6 against the ILNPv6 at the same frame. As shown, there are several types of visual noise artifacts that could be introduced when the packets carrying the H.264 encoded clip are lost.

There are some block noises observed in Figure (b) with MIPv6; the image is jagged with two frames mixed into each other in chunks, while ILNPv6 shows a clean still image in Figure (a). Figure (d), from MIPv6 result, shows a residual key-frame from the previous scene with overlapping fragments of the following scene, with the outline of the part of the correct frame as shown in Figure (c) with ILNPv6 showing on top of the previous scene. A robotic arm and a needle, which were moving during those particular frames, were vaguely showing on top of the frozen frame. Meanwhile, the ILNPv6 result, shown in Figure (c), shows a clean frame without artifacts. Figure (f) displays an incorrectly frozen frame with MIPv6; while with ILNPv6 in Figure (e), the scene progressed correctly.

Aggregate results

The following shows the aggregated results of the video quality metrics SSIM and PSNR. Figure 7.8 shows 0 ms added RTT scenario results. With 0 ms added RTT scenarios, with both clip types, at all resolutions, SSIM was significantly lower with MIPv6. The delivery rate on MIPv6 was significantly lower, similar to iperf2 UDP scenario from the continuous mobility scenario.

Figure 7.9 shows the SSIM and delivery rate from the 20 ms delay scenario. Similar to the 0 ms results, the 20 ms also shows a significant difference between MIPv6 and ILNPv6. The loss of packets in MIPv6 remains more significant and the SSIM was lower. The same trend continued with 200 ms results, shown in Figure 7.10. MIPv6 experienced much lower mean SSIM in both clips and resolutions.

Table 7.3 shows PSNR results. Each cell is colour coded by mean ratio of ‘infinite’ PSNR frames over the thirty runs. ILNPv6 runs had a much higher ratio of frames with ‘infinite’ PSNR values, in other words, ‘*perfect*’ PSNR value frames. In addition, $PSNR_{\min}$ also remained higher than MIPv6 in all scenarios, suggesting that the ‘worst case’ quality loss on average was better than the MIPv6 results. MIPv6 runs resulted in a much larger max–min PSNR value difference,



Figure 7.7: A series of frames captured after converting the received MPEG-TS files to raw YUV 4:2:0 files were used for analysis using VQMT. Images on the left column are frames captured from ILNP results. Images on the right column are frames captured from MIPv6 runs. **(a) and (b)** are the 2256th frame of the recorded clip. Frame (b), from the MIPv6 result, shows clear block noise and overall delay in image transition. **(c) and (d)** are the 1089th frame into the clip. Frame (d) from MIPv6 shows clear colour noise and block noise. Note that the part of the robotic arm with a needle similar to the one seen in (c) is visible, which is from the expected frame. The more dominant image in frame (d) is a man with glasses from the previous scene. **(e) and (f)** are the 1325th frame — they were mismatched. Frame (f) is visually not impaired, but it is not the expected frame at that time, which should be what (e) shows. Upon playing back the MPEG2-TS file, at the same timecode, frame (e) was displayed with the ILNP clip, but for the MIPv6 clip, the player displayed the identical frame shown in (f) for multiple seconds.

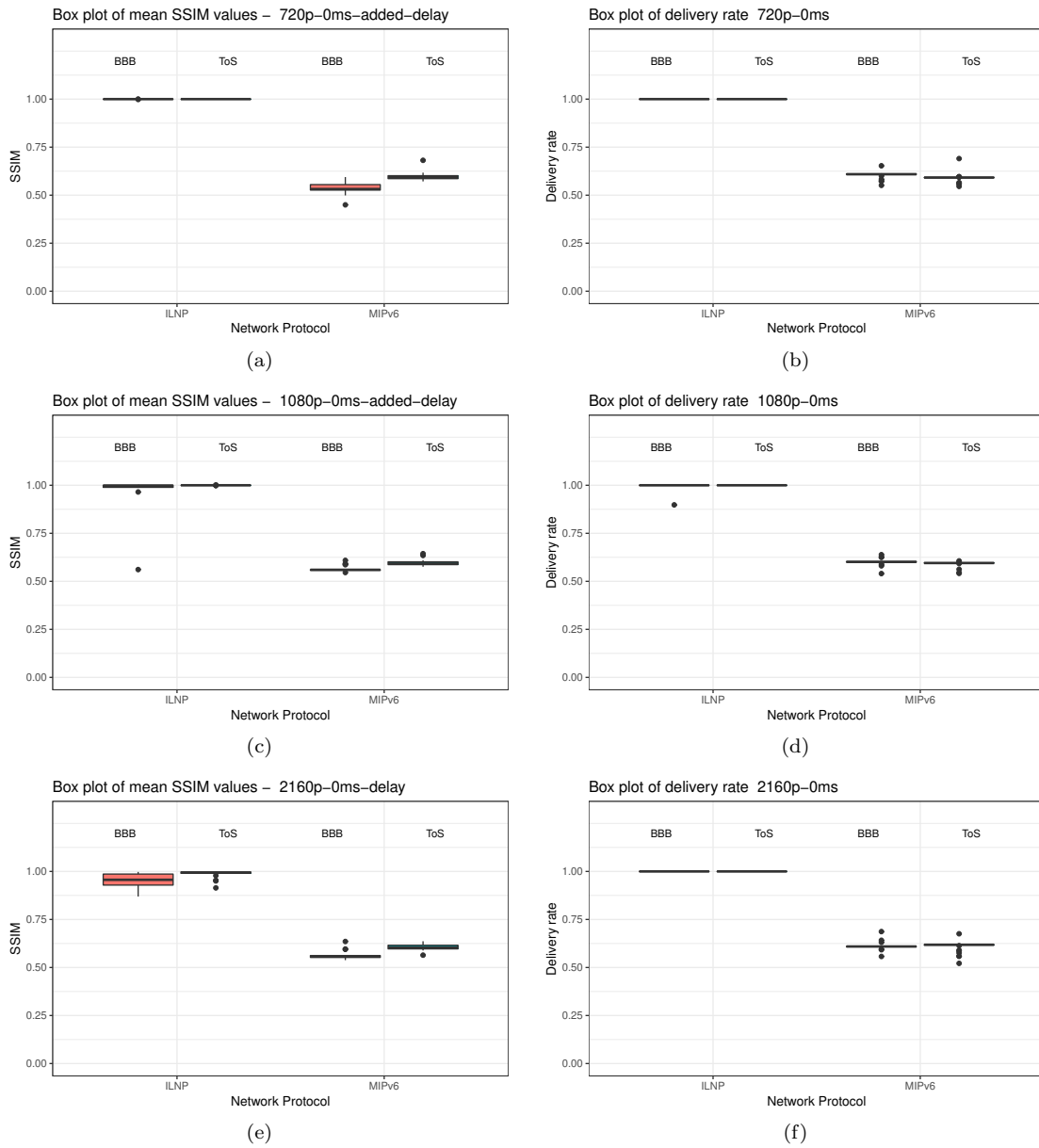


Figure 7.8: Box plots of the mean SSIM and delivery rate values observed for the 120s clip streamed over MiPv6 and ILNP, recorded at the MN.ToS refers to Tears of Steel, a live action sci-fi movie with 3D effects; BBB refers to Big Buck Bunny, a 3D animation movie. An SSIM value of 1.0 indicates being identical to the reference image. The SSIM values are calculated frame by frame against the original clip used for streaming and the mean SSIM is calculated by taking the average throughout a run. The delivery rate values are calculated by observing the RTP sequence numbers from the packet captures.

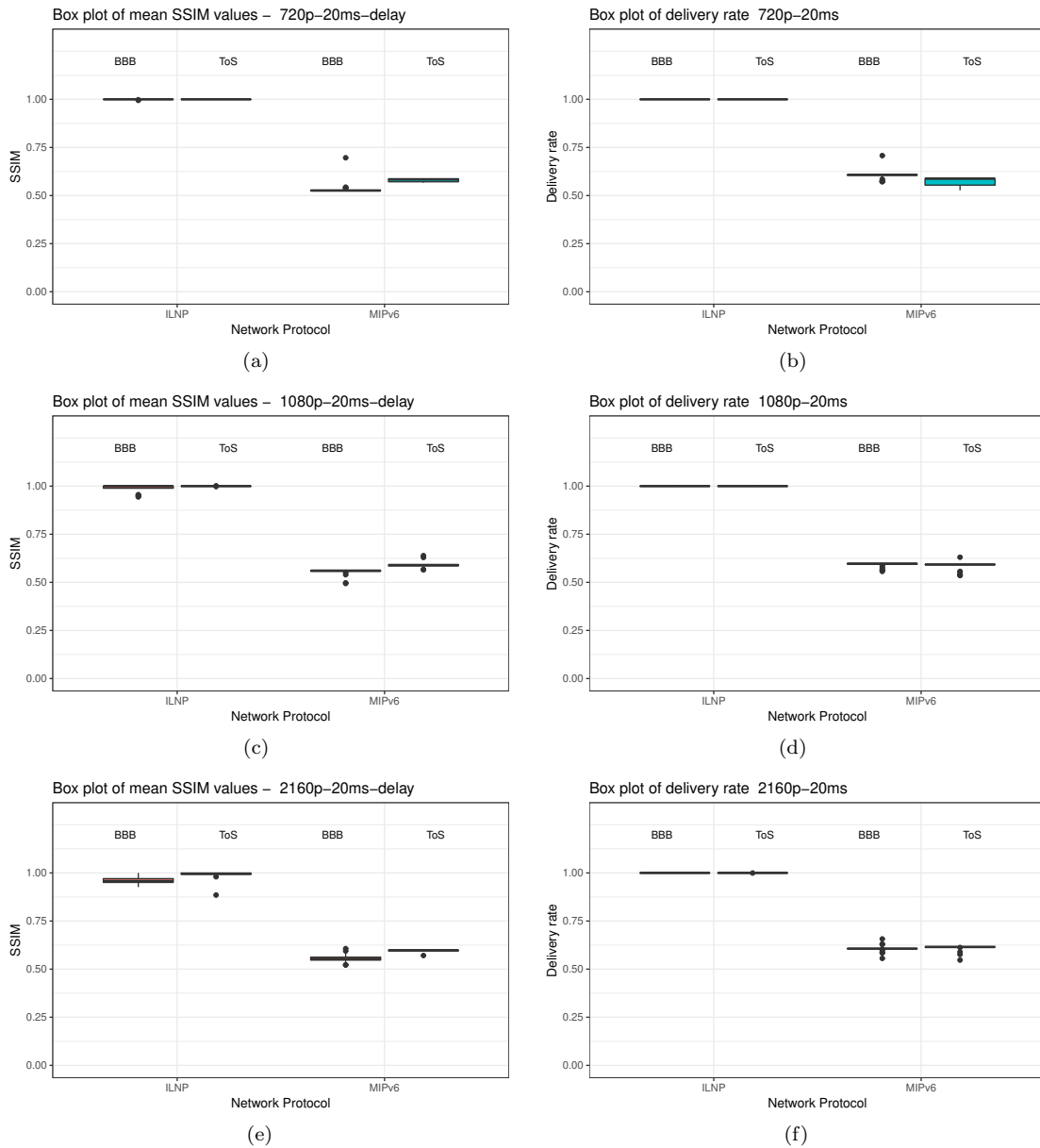


Figure 7.9: Box plots of the mean SSIM and delivery rate values observed for the 120s clip streamed over MIPv6 and ILNP, recorded at the MN. The network has an additional 20ms RTT between the MN and the CN to emulate the wider topological/physical distance between them. ToS refers to Tears of Steel, a live action sci-fi movie with 3D effects; BBB refers to Big Buck Bunny, a 3D animation movie. An SSIM value of 1.0 indicates being identical to the reference image. The SSIM values are calculated frame by frame against the original clip used for streaming and the mean SSIM is calculated by taking the average throughout a run. The delivery rate values are calculated by observing the RTP sequence numbers from the packet captures.

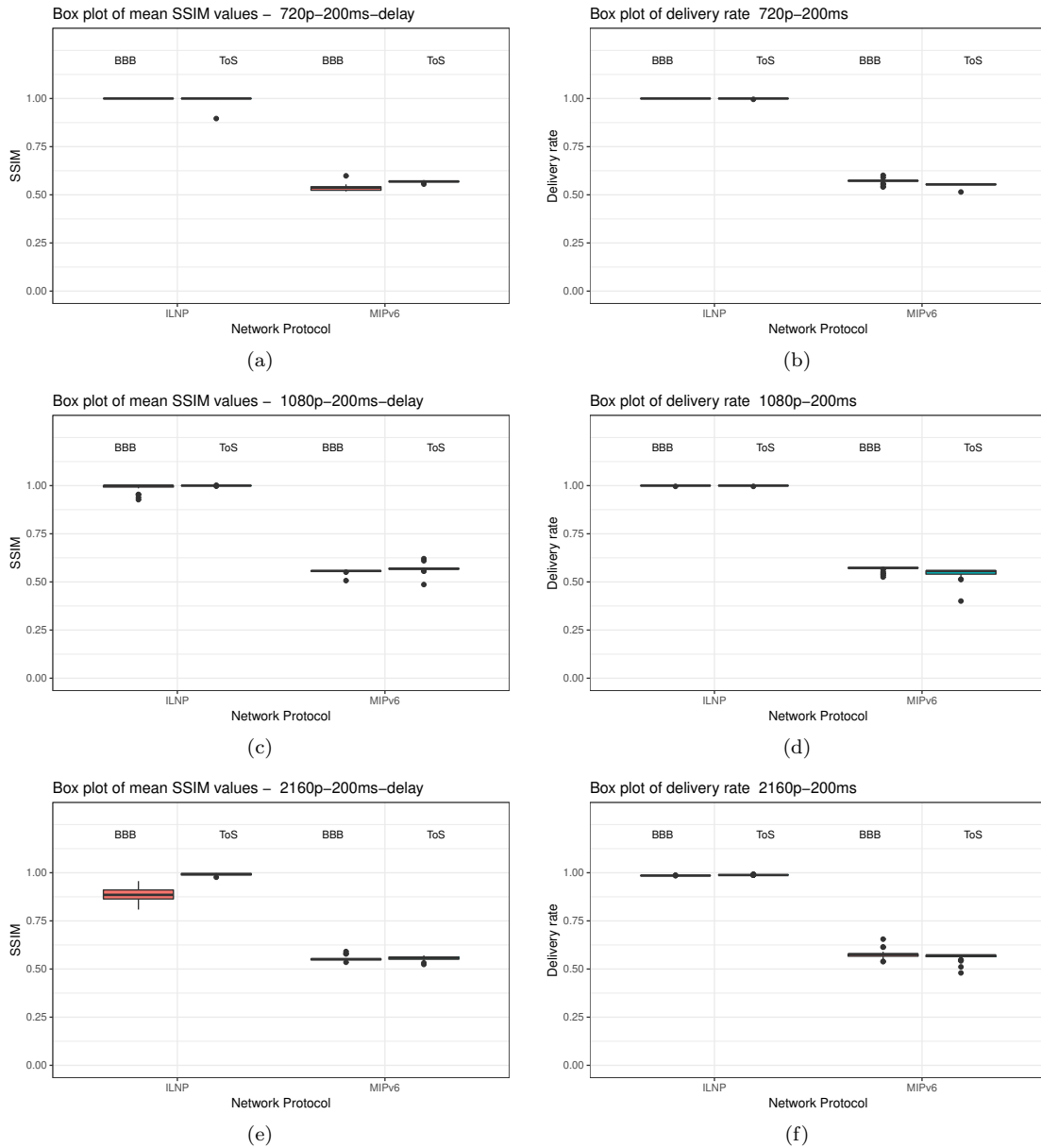


Figure 7.10: Box plots of the mean SSIM and the delivery rate values observed for the 120s clip streamed over MIPv6 and ILNP, recorded at the MN. The network has an additional 200ms RTT between the MN and the CN to emulate the wider topological/physical distance between them. ToS refers to Tears of Steel, a live action sci-fi movie with 3D effects; BBB refers to Big Buck Bunny, a 3D animation movie. An SSIM value of 1.0 indicates being identical to the reference image. The SSIM values are calculated frame by frame against the original clip used for streaming and the mean SSIM is calculated by taking the average throughout a run. The delivery rate values are calculated by observing the RTP sequence numbers from the packet captures.

resulted in jitters in the visual quality observed, so the QoE varied a lot more compared to ILNPv6.

7.6 ‘Real-time’ video transport experiment with multihoming mobility duality

In Chapter 6, ILNPv6 showed iperf2 UDP can function over mobility-multihoming duality. In this section, ILNPv6 mobility-multihoming functionality is used with RTP/UDP payload and evaluated.

7.6.1 Scenario

The ‘movement’ scenario is the same as Section 6.3.1 in Chapter 6; the additional interfaces are set active one by one until all are active, then removed one by one until only one is left active, repeating this again to ‘return’ to the original starting point network. See Figure 4.5 on page 62, which illustrates the MN’s movement; arrows marked with 1 and 2 indicate the order of the movement the MN experienced. Unlike the mobility-multihoming duality iperf scenario, RTP/UDP payload is used instead.

7.6.2 Results

Typical runs

The following Figures 7.11 and 7.12 present a ‘typical run’ for mobility-multihoming duality scenario. Plots show the throughput and RTP sequence numbers observed over the duration; and observed SSIM of the each frames.

Aggregated results

The following shows aggregated results. The plots in Figure 7.13 shows the SSIM observed at the different resolution and added RTT configurations. The results show a consistent median SSIM at near 1 for all scenarios, showing overall ‘*perfect*’ image for majority of the duration.

Scenario		Big Buck Bunny	Tears of Steel
Delay	Format		
0ms	720p	$\langle 0.999, 26.5, 4.01, \frac{2}{30} \rangle$	$\langle 1.00, \text{N/A}, \text{N/A}, \frac{0}{30} \rangle$
	1080p	$\langle 0.940, 20.5, 11.5, \frac{19}{30} \rangle$	$\langle 0.997, 32.3, 8.64, \frac{5}{30} \rangle$
	2160p	$\langle 0.920, 13.7, 21.0, \frac{30}{30} \rangle$	$\langle 0.847, 15.5, 48.7, \frac{30}{30} \rangle$
20ms	720p	$\langle 0.998, 21.9, 2.95, \frac{2}{30} \rangle$	$\langle 1.00, \text{N/A}, \text{N/A}, \frac{0}{30} \rangle$
	1080p	$\langle 0.971, 22.1, 3.22, \frac{15}{30} \rangle$	$\langle 0.998, 31.7, 8.52, \frac{4}{30} \rangle$
	2160p	$\langle 0.857, 13.7, 24.7, \frac{29}{30} \rangle$	$\langle 0.930, 17.2, 32.5, \frac{30}{30} \rangle$
200ms	720p	$\langle 1.00, \text{N/A}, \text{N/A}, \frac{0}{30} \rangle$	$\langle 0.981, 8.70, 57.1, \frac{1}{30} \rangle$
	1080p	$\langle 0.970, 20.9, 9.31, \frac{12}{30} \rangle$	$\langle 0.995, 31.1, 7.84, \frac{8}{30} \rangle$
	2160p	$\langle 0.695, 10.4, 24.3, \frac{30}{30} \rangle$	$\langle 0.933, 14.0, 48.5, \frac{30}{30} \rangle$

(a)

Scenario		Big Buck Bunny	Tears of Steel	Scale
Delay	Format			
0ms	720p	$\langle 0.293, 7.63, 38.5, \frac{30}{30} \rangle$	$\langle 0.455, 7.50, 50.1, \frac{30}{30} \rangle$	1.00
	1080p	$\langle 0.368, 8.19, 48.8, \frac{30}{30} \rangle$	$\langle 0.454, 7.46, 61.9, \frac{30}{30} \rangle$	0.90
	2160p	$\langle 0.348, 7.98, 48.9, \frac{30}{30} \rangle$	$\langle 0.431, 7.35, 66.9, \frac{30}{30} \rangle$	0.80
20ms	720p	$\langle 0.307, 7.65, 39.1, \frac{30}{30} \rangle$	$\langle 0.427, 7.70, 49.0, \frac{30}{30} \rangle$	0.70
	1080p	$\langle 0.358, 7.89, 51.1, \frac{30}{30} \rangle$	$\langle 0.423, 7.61, 57.7, \frac{30}{30} \rangle$	0.60
	2160p	$\langle 0.337, 7.96, 47.6, \frac{30}{30} \rangle$	$\langle 0.435, 7.35, 65.1, \frac{30}{30} \rangle$	0.50
200ms	720p	$\langle 0.288, 7.61, 36.3, \frac{30}{30} \rangle$	$\langle 0.409, 7.62, 37.9, \frac{30}{30} \rangle$	0.40
	1080p	$\langle 0.354, 8.02, 41.5, \frac{30}{30} \rangle$	$\langle 0.400, 7.61, 44.7, \frac{30}{30} \rangle$	0.30
	2160p	$\langle 0.322, 7.22, 45.8, \frac{30}{30} \rangle$	$\langle 0.378, 7.41, 60.4, \frac{30}{30} \rangle$	0.20

(b)

Table 7.3: Tables show PSNR results for ILNP (a) and MIPv6 (b) in continuous-mobility scenarios. Each cell contains a tuple of the following metrics: (1) the mean ratio of infinite PSNR frames (this value is what is used for the table colouring, as represented by the scale on the right), (2) the mean minimum PSNR value, (3) the difference between the mean maximum and minimum PSNR value, and (4) the number of runs with finite PSNR value frame/number of runs. The mean ratio of infinite PSNR frames indicates the mean ratio of ‘perfect’ frames in each run, the higher the better. The mean minimum PSNR indicates the average ‘worst case scenario’, indicating the worst case quality metric on average. The difference between the mean maximum and minimum PSNR value indicates the average size of fluctuation in the video quality. (4) indicates how many runs out of the total number of runs were used in calculating the first three elements of the tuple.

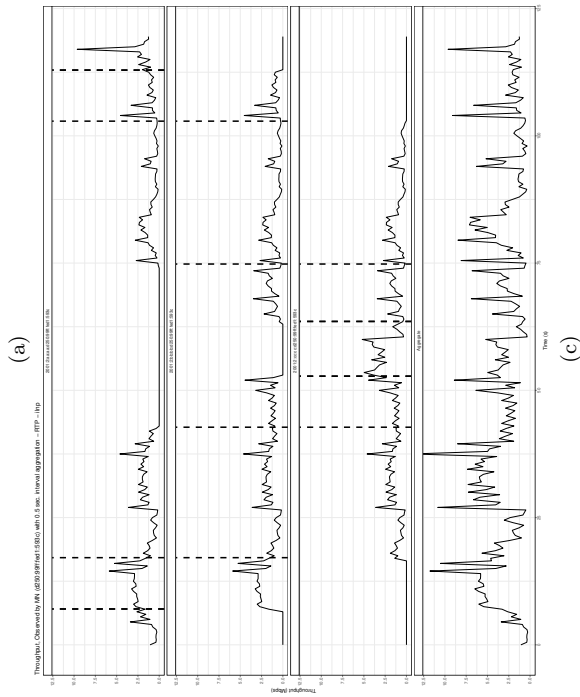
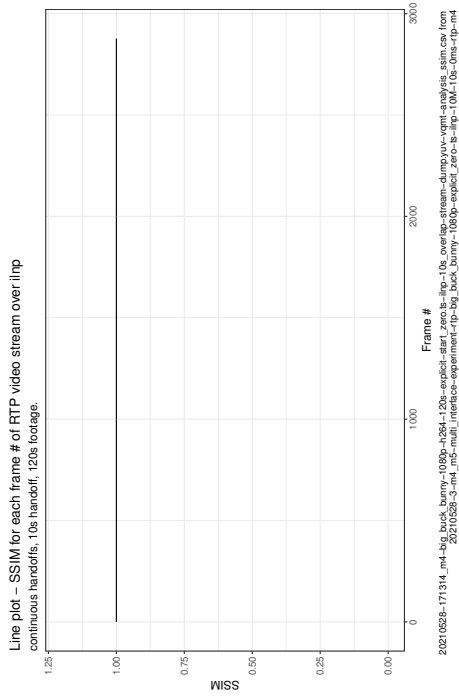
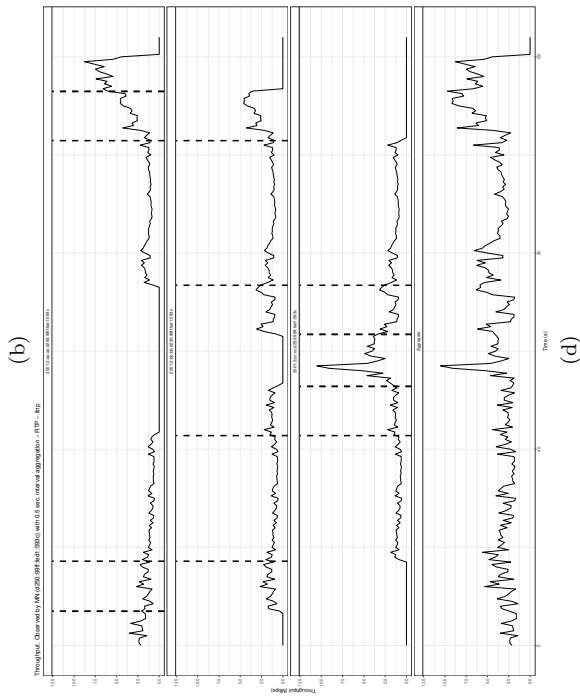
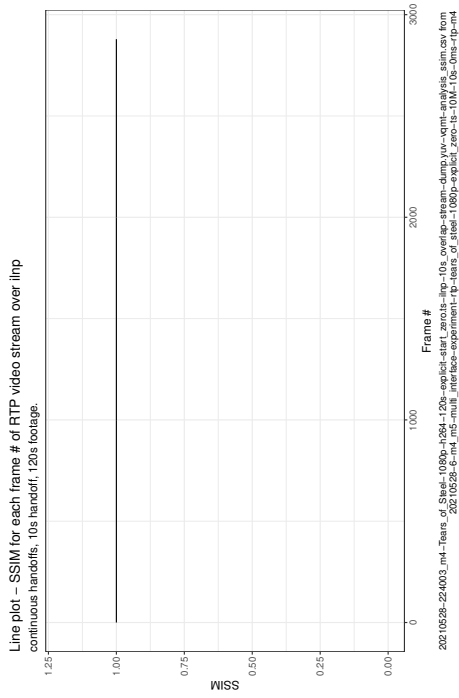


Figure 7.11: Example plots showing the SSIM per frame and throughput over time for the two clips: Tears of Steel and Big Buck Bunny. (a) and (c) show the result from Big Buck Bunny, and (b) and (d) show the result from Tears of Steel. They are both 1080p clips with no added delay in the network. The top plot shows the SSIM per frame — SSIM is the structural similarity of the captured clips to the original clip. The flat line at 1.0 indicates that the received image is identical to the original in the context of SSIM. The facet throughput plots show the throughput received at the respective three L64s on three interfaces in the top three plots, with the aggregate of them at the bottom of the facet plots. Dashed lines show the LU/LU-Ack received/sent from the respective interface. The aggregate throughputs here are very similar to the reference static IPv6 single home run shown in Figure 7.7.

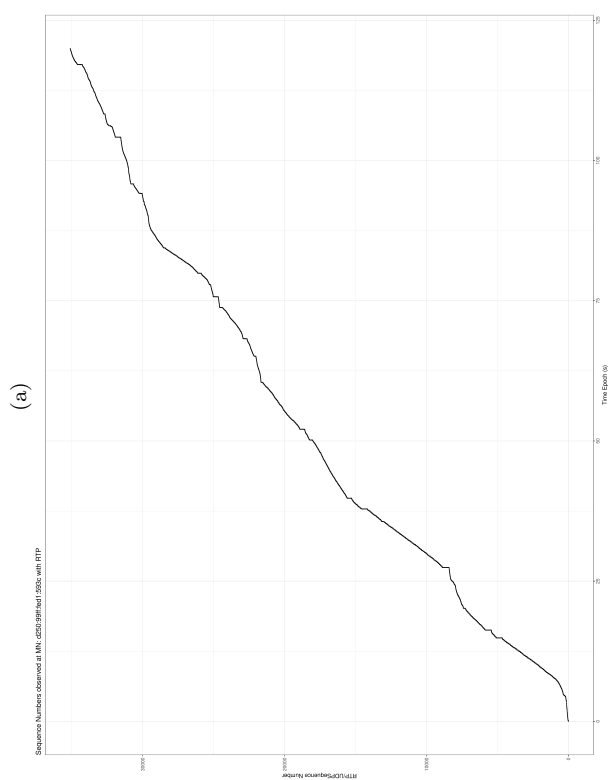
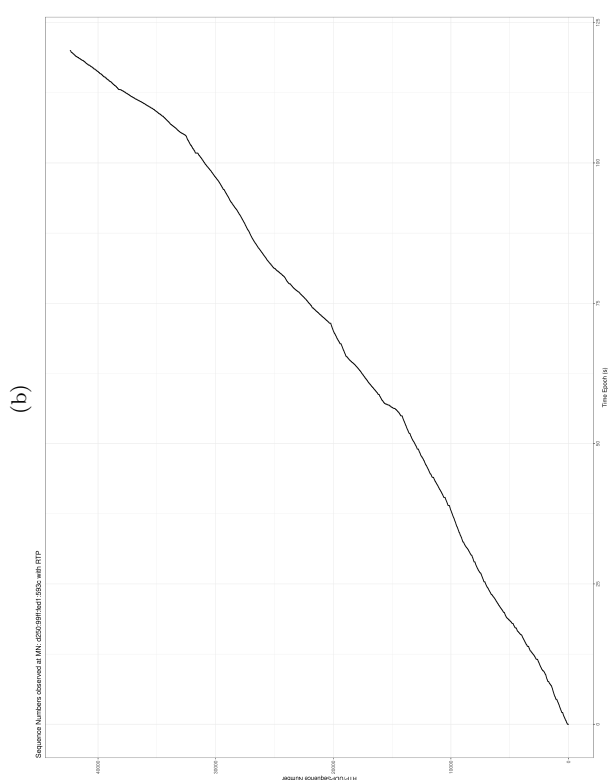
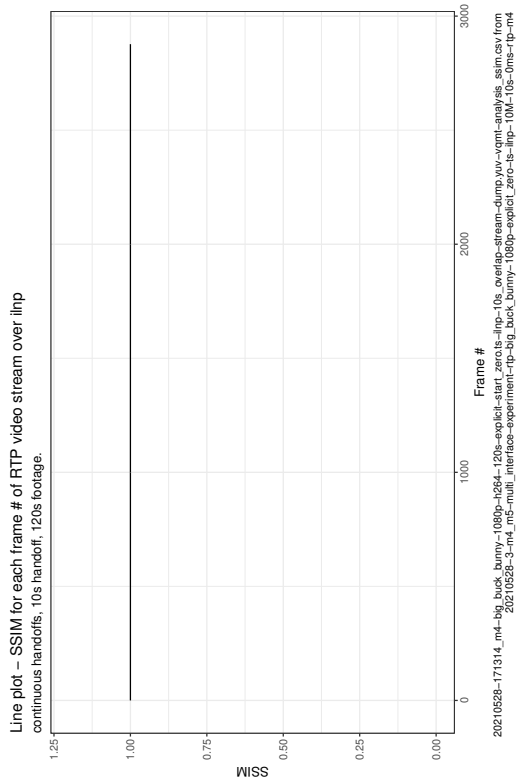
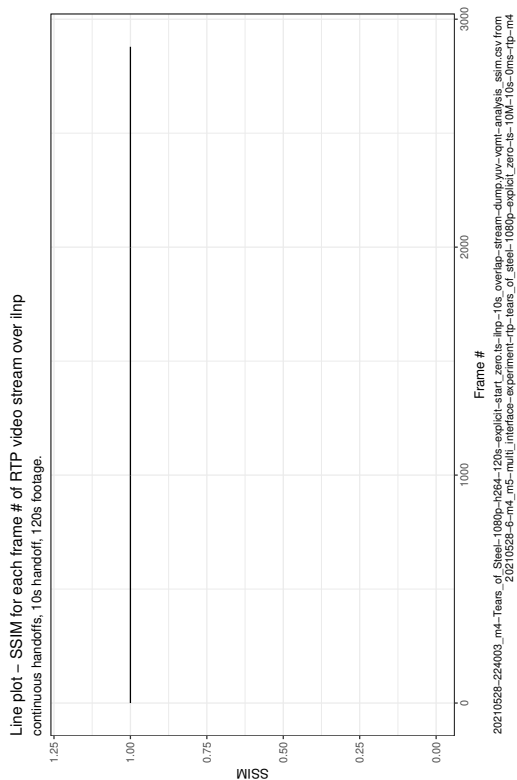


Figure 7.12: Example plots showing the SSIM per frame and the sequence numbers observed over time for the two clips: Tears of Steel and Big Buck Bunny. They are both at 1080p resolution. Plots (a) and (c) show results from Big Buck Bunny and (b) and (d) show results from Tears of Steel. (a) and (b) show the SSIM per frame with both cases indicating 1.0 throughout the flow. In the context of SSIM, 1.0 indicates that the original and the received images are identical. The sequence numbers are observed from the RTP packet header. The continuous plot without any gaps indicates that there were no significant loss in data transfer. Also, there are no noticeable amount of misordering visible in the plot, which would have been indicated by diverging 'lines'.

	ILNP	MIPv6	—	IPv6 ¹
720p				
Tears of Steel	2.01 Mbps	1.23 Mbps	—	2.04 Mbps
Big Buck Bunny	1.89 Mbps	1.18 Mbps	—	1.93 Mbps
1080p				
Tears of Steel	3.72 Mbps	2.27 Mbps	—	3.71 Mbps
Big Buck Bunny	3.06 Mbps	1.91 Mbps	—	3.13 Mbps
2160p				
Tears of Steel	15.8 Mbps	9.98 Mbps	—	16.1 Mbps
Big Buck Bunny	8.82 Mbps	5.59 Mbps	—	8.99 Mbps

1. Reference-only, single-run measurement using fixed single connection.

Table 7.4: A table showing the mean throughput for each clip in 0ms continuous-mobility scenarios. The average throughput is derived from 30 runs by dividing the total payload length per run received by the duration of the run. As MIPv6 handoffs result in multiple-second loss of throughput, the overall throughput achieved is much lower than ILNPv6, as ILNPv6 provides smooth handoff with near-zero loss. The final column is a reference-only measurement without mobility handoffs. ILNPv6, while providing continuous handoffs, achieves very similar overall throughput as the reference IPv6-only throughput.

In Tears of Steel (ToS), while some of the 2160p results experience a slightly lower mean SSIM, they were mostly caused by momentarily low SSIM, which lowered the mean minimum lower. For all other resolutions, the minimum also remained at 1 or very close to 1 — within 0.1. In Big Buck Bunny (BBB), all of the median remained at 1, where mean SSIMs remained at 1 and close to 1 — within 0.15 from 1. This indicates, similarly to mobility, for majority of the duration, the observed image was *‘perfect’*. However the low minimum for some of the cases has pulled the mean SSIM down. In BBB, the lower minimum and the mean SSIM was observed, overall, more than ToS.

Figure 7.14 shows the RTP sequence number stats. The misordering rate increased as the added RTT became larger, which is expected and they were uniform regardless of the clips streamed. While the general trend of the impacts of added RTT was the same in both clips, the scale of impacts were different, indicating the different types of footage may be impacted differently to the way underlying network conditions differ.

Similar to the continuous mobility scenario results, ILNPv6 delivered smooth handoffs. As expected from the results from Chapter 6, loss rates are at 0 for most cases, while misordering rate were less than 10%. The median SSIM remained above 0.9 for all scenarios.

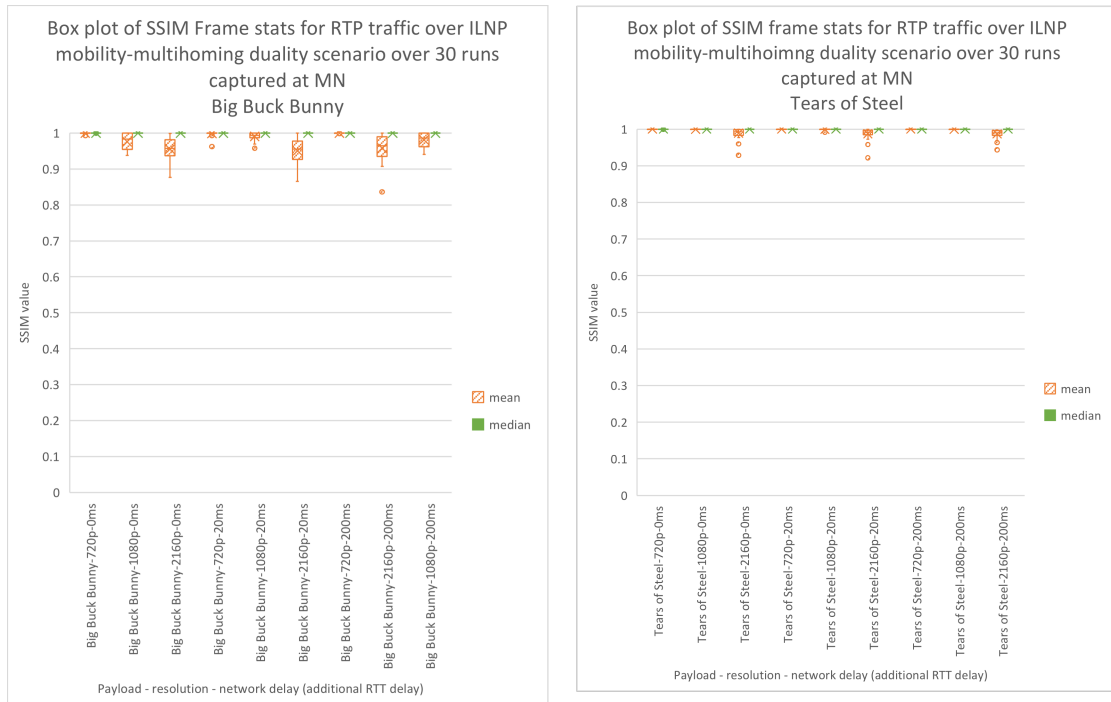


Figure 7.13: Boxplots showing the SSIM stats for the mobility-multihome duality RTP flows. As shown, both the median and the mean stay close to 1.0, indicating little to no loss of quality for a large proportion of time. In some cases, especially for Big Buck Bunny, for longer RTT, some quality degradation were observed, which are shown as slightly lower mean SSIM. However, provided that both the median and the mean still remain at near 1.0, only a very small portion of the clip experienced quality degradation. The × indicates the mean.

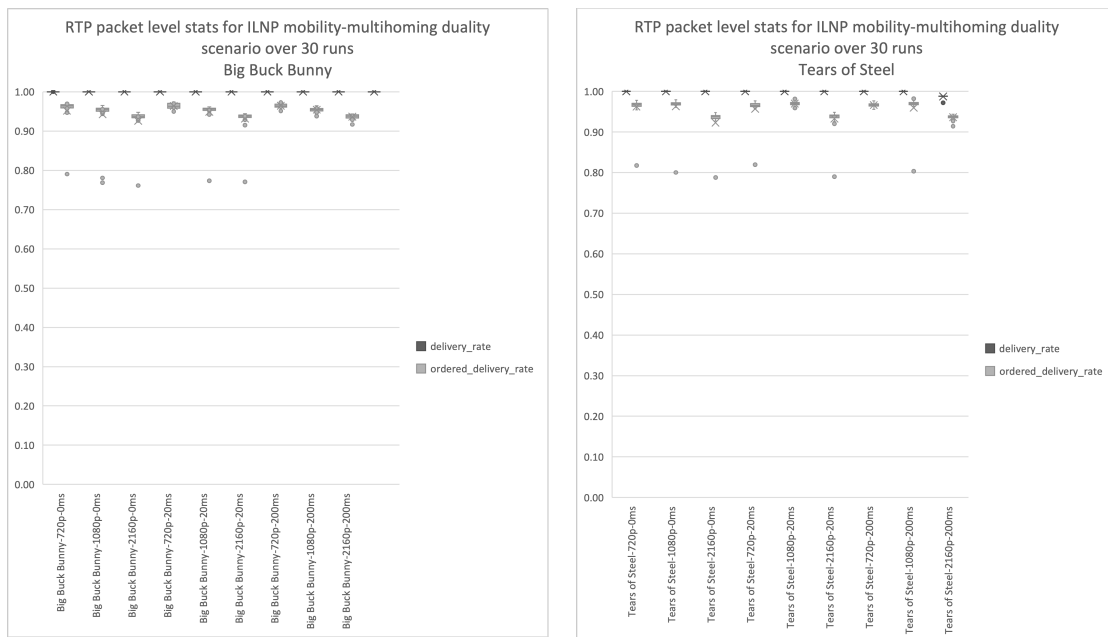


Figure 7.14: Boxplots showing the RTP packets stats for mobility-multihome duality RTP flows. As the RTT increased, the misordering observed increased. The loss remained little to none. As these were RTP/UDP flows, no duplicates were observed as expected. The × indicates the mean.

Scenario		Big Buck Bunny	Tears of Steel	Scale
Delay	Format			
0ms	720p	$\langle 0.991, 26.5, 17.2, \frac{8}{30} \rangle$	$\langle 1.00, \text{N/A}, \text{N/A}, \frac{0}{30} \rangle$	1.00
	1080p	$\langle 0.930, 15.4, 8.84, \frac{23}{30} \rangle$	$\langle 0.991, 28.3, 7.00, \frac{15}{30} \rangle$	0.90
	2160p	$\langle 0.846, 13.8, 24.6, \frac{30}{30} \rangle$	$\langle 0.929, 17.7, 44.7, \frac{30}{30} \rangle$	0.80
20ms	720p	$\langle 0.996, 18.6, 11.8, \frac{3}{30} \rangle$	$\langle 1.00, \text{N/A}, \text{N/A}, \frac{0}{30} \rangle$	0.70
	1080p	$\langle 0.952, 19.0, 14.2, \frac{21}{30} \rangle$	$\langle 0.991, 28.6, 6.74, \frac{15}{30} \rangle$	0.60
	2160p	$\langle 0.841, 14.1, 30.8, \frac{29}{30} \rangle$	$\langle 0.938, 15.5, 35.3, \frac{27}{30} \rangle$	0.50
200ms	720p	$\langle 0.994, 28.1, 12.9, \frac{3}{30} \rangle$	$\langle 1.00, \text{N/A}, \text{N/A}, \frac{0}{30} \rangle$	0.40
	1080p	$\langle 0.947, 17.2, 4.96, \frac{22}{30} \rangle$	$\langle 0.994, 28.1, 6.42, \frac{11}{30} \rangle$	0.30
	2160p	$\langle 0.861, 14.5, 25.5, \frac{29}{30} \rangle$	$\langle 0.935, 13.7, 49.1, \frac{30}{30} \rangle$	0.20
				0.10

Table 7.5: A table showing PSNR results for the ILNPv6 mobility-multihoming duality scenario. Each cell contains a tuple of the following metrics: (1) the mean ratio of infinite PSNR frames (this value is what is used for the table colouring, as represented by the scale on the right), (2) the mean minimum PSNR value, (3) the difference between the mean maximum and minimum PSNR value, and (4) the number of runs with finite PSNR value frame/number of runs. The mean ratio of infinite PSNR frames indicates the mean ratio of ‘perfect’ frames in each run, the higher the better. The mean minimum PSNR indicates the average ‘worst case scenario’, indicating the worst case quality metric on average. The difference between the mean maximum and minimum PSNR value indicates the average size of the fluctuation in the video quality. (4) indicates how many runs out of the total number of runs were used to calculate the first three elements of the tuple.

	ILNP	—	IPv6 ¹
720p			
Tears of Steel	2.04 Mbps	—	2.04 Mbps
Big Buck Bunny	1.93 Mbps	—	1.93 Mbps
1080p			
Tears of Steel	3.76 Mbps	—	3.71 Mbps
Big Buck Bunny	3.13 Mbps	—	3.13 Mbps
2160p			
Tears of Steel	16.2 Mbps	—	16.1 Mbps
Big Buck Bunny	8.98 Mbps	—	8.99 Mbps

1. Reference-only, single-run measurement using fixed single connection.

Table 7.6: A table showing the mean throughput for each clip in 0ms mobility-multihoming duality scenarios. The average throughput is derived from 30 runs by dividing the total bits received in a run by the duration. The final column is a reference-only measurement without mobility handoffs. ILNPv6, while providing continuous multihomed soft-handoffs, achieves very similar overall throughput as the reference IPv6-only throughput.

7.7 Discussion

7.7.1 ‘Real-time’ video transfer as a *real-life, real-time* application and its design space

In the current Internet, mobility and multihoming require significant effort within the application itself. This includes middlewares connecting different applications. Applications have to come up with individual mechanisms to manage addresses and handle changes in connectivity. However, as shown in this chapter, ILNPv6 can enable both mobility and multihoming without modification to the application binary, nor new special transport protocol, and without requiring modifications to the network infrastructures. In the case of mobility, ILNPv6 enabled handoffs without little to no loss, while the current Internet Engineering Task Force (IETF) mobility solution, MIPv6, could not provide smooth handoffs, impacting the QoE. While it is true that applications will have to continue to manage loss and other Quality of Service (QoS) impacts from underlying network technologies, the results showed that with ILNPv6, mobility and multihoming, and address management, can be handled at the network layer, without changing the protocol behaviour, and without introducing a significant amount of QoS degradation.

The evaluations in Chapters 5–6 were conducted with unmodified commonly used open-source software. Sections 7.5 and 7.6 showed RTP functioning over ILNPv6 without any modifications to the applications running on the end-hosts. This was realised, once again, without modifying commonly used open-source software `ffmpeg` and `vlc`. Real-time applications are a challenging use case for mobility and multihoming, as loss and misordering impact the QoE directly as shown in Section 7.5. However, the results showed that ILNPv6 could deliver near-perfect performance while enabling continuous mobility or mobility-multihoming duality. Again, this was achieved using off-the-shelf commercial networking equipment, with commonly used open-source software, with off-the-shelf motherboards on the end-hosts, and with simple standard IPv6 network — all without any knowledge of ILNPv6.

This also demonstrates that other applications that were unable to implement mobility or multihoming could function in such a scenario, potentially expanding their use cases. This allows applications to focus on other aspects of real-time networked applications, such as adapting to

QoS variations of the underlying connectivities (which is already needed to a certain degree for single-homed hosts), without needing to implement address management or signalling mechanisms required for mobility and multihoming.

7.7.2 ‘Real-time’ video scenario with pre-recorded clips

In ‘real world’ ‘real-time’ video applications, such as video calls and live video broadcast, the video comes from a camera capturing the image in real-time. While it is true that a 4K UHD webcam may be used as a source to emulate the real-time aspect, for data-collection and reproducibility, pre-recorded clips were used instead. In order to use a real-time live video from a camera and provide reproducibility, the video has to be recorded at the sender’s side, as well as the receiver’s side, which poses challenges in storage capacity and data processing, especially for a larger resolution. Second, while reproducing real-time video streams with the recorded live videos may yield the same results, it will not completely replicate the environment the system was in, specifically in the context of the processing load on the sender’s side. In fact, that would be identical to the procedure used here: streaming a pre-recorded clip as a real-time video over RTP. Also, streaming pre-recorded clips is a perfectly reasonable potential use case, especially in the context of television broadcast over IP. Although the clip was pre-recorded, formats of the clip were common formats used in real-time video applications. The payload was stored in an MPEG-TS container, which is commonly used with live television broadcast. H.264 is widely used in video call applications as well, such as Microsoft Teams, which is a real-time interactive application.

Given that H.264 is also used in interactive applications, one could speculate that ILNPv6 could potentially provide good QoE in those scenarios as well. However, to adequately evaluate interactive real-time applications, it would require subjective QoE evaluation with human subjects.

7.8 Summary

Video transmission is a popular application of the Internet. Real-time video transmission is especially challenging due to its sensitiveness to changes in QoS, which can result in problems such

as loss, which would affect the QoE. Results showed that ILNPv6 was able to provide *true continuous mobility*, providing consistently better QoE than MIPv6. In addition, ILNPv6 provided *dynamic mobility-multihoming duality*, also providing excellent results with QoE metrics such as PSNR and SSIM. In this chapter, ILNPv6 not only outperformed MIPv6, but it also enabled *true end-to-end mobility-multihoming duality* for existing, popular, open-source applications without any modifications, without modifying any part of the network infrastructures, without modifying standard IPv6 unicast routing, and without additional infrastructures like Home Agent (HA), middlebox, or proxy; this was all done directly between just the two participating hosts, demonstrating a truly dynamic and flexible communication capability.

Chapter 8

Conclusion

8.1 Contributions

As stated in Chapter 1, the thesis statement is as follows:

Mobility and multihoming duality can be realised through an end-host solution, without requiring changes to core infrastructures or existing applications.

Chapter 4 described the problem space, testbeds, evaluation scenarios, methods, and common metrics. It presented the evaluation testbed, which was used in later chapters to demonstrate two key points:

- (1) ILNPv6 does not require changes to core infrastructures.
- (2) ILNPv6 is an end-host-only solution.

(1) is demonstrated by the testbed network, which consists of commercial off-the-shelf network equipment, without any modification or custom packages installed, operating as a simple IPv6 network with static packet forwarding. (2) is demonstrated by the lack of any special network entities in the network, unlike MIPv6, which requires the HA.

Chapter 5 showed the contributions to the ILNPv6 system design and the implementation to enable continuous mobility. The results from a comparative evaluation against MIPv6 were presented, showing that ILNPv6 had a superior performance during a communication session over multiple handoffs.

Chapter 6 introduced the novel mobility-multihoming duality mechanism implemented in ILNPv6. Changes to control plane logic and relevant packet processing were shown. The empirical evaluation results were presented showing a dynamic use of connectivities, using the mobility-multihoming duality mechanism implemented in ILNPv6. The results presented both TCP and UDP transport functioning over the mobility-multihoming duality mechanism with near-zero gratuitous loss and smooth changes to the use of connectivities.

Chapter 7 presented *real* application level performance analyses. Specifically, QoE analyses of a real-time video application was carried out using two scenarios. For the continuous mobility scenario, a comparative evaluation against MIPv6 was carried out showing ILNPv6's much smoother handoffs, enabling much better QoE across different delay profiles between the MN and the CN, across three different resolutions and two types of clips. For mobility-multihoming duality, the empirical evaluation of QoE metrics presented near-zero gratuitous impact of the handoffs while still enabling mobility-multihoming duality for legacy applications, over an unmodified, plain IPv6 testbed network.

8.2 Discussion

8.2.1 Lack of existing network layer host mobility-multihoming duality solutions

In Chapters 6–7, the mobility-multihoming duality scenario did not include a comparison. This is due to the fact that as of June 2020, there were no candidates for direct comparison. This continues to be the case in October 2021.

As discussed in Chapters 2–3, most solutions approached either of the mobility or multihoming as the primary function. Some, have then added the other function retrospectively into the specification or have indicated that it can be realised with some additional mechanism.

Level 3 Multihoming Shim Protocol for IPv6 (SHIM6) does define both mobility and multihoming using SHIM6; however, it is not clear whether they can function simultaneously, nor are there any implementations to verify as such. Similarly, Host Identity Protocol (HIP) RFCs

define both mobility and multihoming in RFC8046 [40] and RFC8047 [41], but no known implementations were available to test with at the time.

While there are transport-layer solutions claiming to have mobility-multihoming duality mechanisms, they are not appropriate for *side-by-side* comparison. Because they operate at the transport-layer, unlike ILNPv6, they cannot support new transport-layer protocol, and requires changes to the application. ILNPv6 makes use of the existing socket APIs, allowing applications to function unmodified. In this study, neither `iperf2`, `ffmpeg`, nor `vlc` were altered in any way, shape, or form to function over ILNPv6.

8.2.2 Effects on TCP congestion control

Previously, the design of TCP congestion control algorithms assumed a single path, as TCP state tuple assumes single-homed connectivity, concerned with one single path characteristics at a given time. However, when multiple network connectivities are used, such assumption no longer holds — MultiPath-TCP (MP-TCP) required a dedicated congestion control mechanism to handle multi-path effects, such as misordering and variations to RTT. With ILNPv6 enabling mobility-multihoming duality at the network layer, not only the QoS change as the host changes its connectivity, the host may also be utilising multiple connectivities simultaneously with legacy applications, with existing transport protocols, in situations where QoS will likely be different across different paths. Potentially, with more adaptive TCP congestion control mechanism, mobility-multihoming duality transport with TCP may perform better in challenging environments, utilising all paths in its full potential, which may further improve the performance beyond shown in this work.

8.3 Future work

8.3.1 Path selection and management mechanism

The mobility and multihoming mechanisms in ILNPv6 used and built for the studies in this thesis have a fixed behaviours. That is, the mobility mechanism always selects the ‘newer’ path; the multihoming-mobility scenario mechanism always uses all available paths and distributes load

equally unless the interface is listed in the ‘disabled’ interface list. And the above behaviours do not account for the path capacity or condition. Therefore, if the paths have different throughput capacities, TCP may not fully utilise all of the bandwidth available to the host, and UDP communication may experience loss. Algorithms that monitor the condition and adjust the load balancing, or a mechanism that falls back to the previous link when the ‘newer’ path is not the most ‘ideal’ path, may allow the hosts to better utilise the resources available.

8.3.2 Potential privacy enhancements with NID values

Current implementation of ILNPv6 uses the standard IPv6 stateless address autoconfiguration (SLAAC) interface identifier (IID) generation mechanism to generate the Node Identifier (NID) values. However, ILNPv6 can potentially use any of the IID generation mechanisms described in RFC8064 [76].

In addition, ILNPv6 could take privacy further by utilising ephemeral NIDs. The use of a unique NID for each transport session, combined with dynamic host multihoming presented in Chapter 6 — which would involve multiple Locators (L64s) — can greatly increase the effort to track the communication, improving privacy [77].

8.3.3 New transport protocol and ILNPv6

As ILNPv6 operates at the network layer, both old and new transport protocols can operate over ILNPv6 benefiting from its mobility and multihoming feature (and any other additional ILNPv6 features that may be implemented in the future). Though it was not included in the evaluation, for example, as QUIC operates over UDP, it is likely that QUIC may operate over ILNPv6, which removes the need for QUIC to implement address management for mobility mechanisms. In addition, multihoming extension is being developed for QUIC; with ILNPv6, similar to mobility, QUIC multihoming extension could focus on managing the multipath effect instead of the address management. With additional application programming interfaces (APIs), QUIC could take advantage of ILNPv6 further to enable more advanced traffic management. At the time of writing, there are some movement to implement more network features outside of the operating system (OS), such as QUIC and eBPF. If more new transport layer protocol emerged

using UDP as a way to encapsulate packets, similar to QUIC, there is a potential for ILNPv6 to eliminate the need for those protocol to handle address management and signalling. While this is outside the scope of the project, this certainly can be examined by extending the evaluation testbed presented in Chapter 4 to conduct an empirical evaluation.

8.3.4 Enabler of ubiquitous computing

In the space of ubiquitous computing, IP has been considered as one of the barriers to enabling a true ubiquitous computing. As mentioned in Chapter 2, Mark Weiser has described that IP, namely its addressing and use of addresses, prevented his vision from being realised. This thesis showed that ILNPv6 can realise communication between two hosts, utilising an arbitrary number of interfaces, dynamically adding or removing them, without any noticeable loss, over off-the-shelf networking equipment, and with unmodified *‘real’* applications commonly used today. With small changes to the APIs, all of the host-wide movements shown in this thesis, specifically in Chapters 6–7, could be done on a per application, per socket basis. ILNPv6 could be the true enabler of ubiquitous communication, thus the enabler of ubiquitous computing.

Appendix A

ILNPv6 implementation limitation

ILNPv6 is an engineering solution to realise ILNP while maintaining backwards compatibility with IPv6. This particular implementation implemented to Linux 4.9.52 originates from another prototype implementation on Linux 3.9.7. There are several limitations and assumptions that may not cover some of more niche cases. This was partly due to time limitation and to concentrate the effort to more critical part of the development and experiments.

A.1 Multiple interfaces present on the same IPv6 subnet

Current implementation produced in this project does not allow following scenario:

1. With a given subnet 'a', a node is connected via an interface '1'
2. the node connects an interface '2' to the same subnet 'a', while interface '1' is connected

Such action will lead to duplicate address detection (DAD) to fail as this implementation only supports one NID per node. In the IPv6, this is equivalent to having same IID within a subnet. In order for this scenario to behave gracefully, ILNPv6 node should create another NID and assign them accordingly.

If L2 interface bonding is in place, at L3, this is considered a single interface, thus from the operating system perspective, such scenario will be treated as a single virtual interface with single IP address.

A.2 Multiple NIDs

Current implementation does not include a support for multiple Node Identifier (NID). The ILNP, as an architecture, should allow multiple NID on a node. In such scenario, different application should be able to pick a NID. Each application should be able to utilise L64s that are available to the respective NID the application is bound to. However, current architecture does not have a mechanism to migrate to a different NID.

Appendix B

Experimental values

B.1 Introduction

This appendix presents additional data collected in order to determine the RTT values and the overlap duration used in the evaluations presented in this thesis. This is in relation to the Sections 4.5.5–4.5.6.

B.2 Overlap duration

This section shows the distribution of time it took for the interface to become available at the layer-3. Section 4.5.5 describes how the measurements were taken. As mentioned in Section 4.5.5, the measurements were taken over 100 runs. Figure B.1 shows the distribution.

B.3 Round trip time (RTT) selection

RTT across the three paths available to the MN were configured homogenously. Additional latency was added to provide, in total, three RTT configurations. `ping6` program

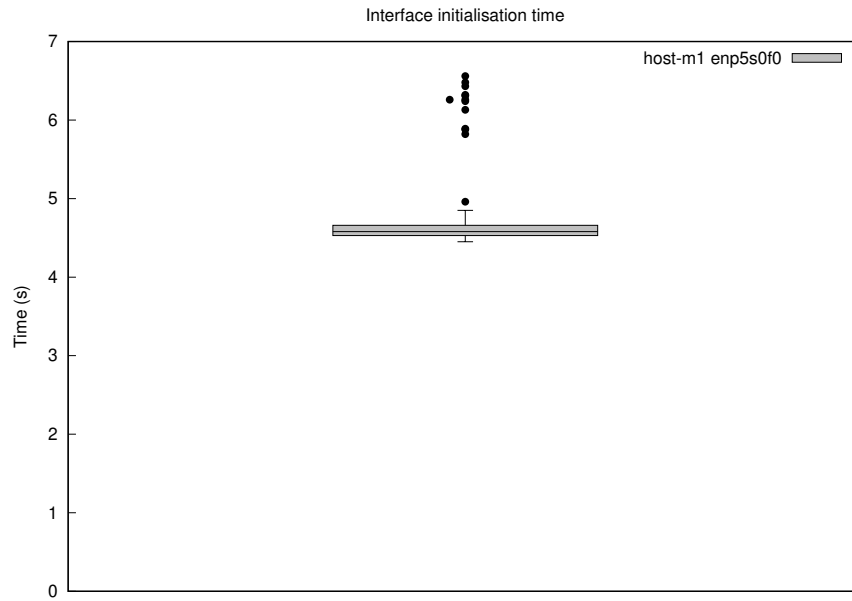


Figure B.1: A boxplot showing the distribution of the time taken for the layer-3 IPv6 connectivity to become available from the time the interface was enabled. The measurement was taken for 100 runs.

B.3.1 ‘National’ RTT traces

To determine the ‘national’ RTT, RTT from a domestic IPv6 network (Sky Broadband — AS5607) in St Andrews to hosts in London, specifically University College London (UCL)’s web server was used to determine the ‘typical’ RTT a host may encounter when communicating with a host across the country.

RTT to the UCL web server — London

```

ryo@Argus:~ $ ping6 www.ucl.ac.uk
PING www.ucl.ac.uk(2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2)) 56 data bytes
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
    icmp_seq=1 ttl=59 time=17.3 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
    icmp_seq=2 ttl=59 time=17.0 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):

```


icmp_seq=3 ttl=59 time=17.2 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=4 ttl=59 time=16.9 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=5 ttl=59 time=19.7 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=6 ttl=59 time=17.5 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=7 ttl=59 time=17.1 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=8 ttl=59 time=17.0 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=9 ttl=59 time=17.2 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=10 ttl=59 time=16.9 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=11 ttl=59 time=17.4 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=12 ttl=59 time=17.2 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=13 ttl=59 time=16.8 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=14 ttl=59 time=16.9 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=15 ttl=59 time=16.9 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=16 ttl=59 time=17.3 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
icmp_seq=17 ttl=59 time=16.7 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):

```

    icmp_seq=18 ttl=59 time=16.6 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
    icmp_seq=19 ttl=59 time=17.2 ms
64 bytes from 2606:4700:4400::6812:27c2 (2606:4700:4400::6812:27c2):
    icmp_seq=20 ttl=59 time=17.2 ms
^C
--- www.ucl.ac.uk ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 47ms
rtt min/avg/max/mdev = 16.616/17.208/19.731/0.640 ms

```

Traceroute result to the UCL web server — London

```

ryo@Argus:~ $ traceroute6 www.ucl.ac.uk
traceroute to www.ucl.ac.uk.cdn.cloudflare.net (2606:4700:4400::6812:27c2)
from 2a02:c7f:2452:fe00:ba27:ebff:feba:acfc, port 33434, from port 43593,
30 hops max, 60 bytes packets
 1  2a02:c7f:2452:fe00:b6fb:e4ff:feb5:a280
    (2a02:c7f:2452:fe00:b6fb:e4ff:feb5:a280)  0.730 ms  0.505 ms  0.712 ms
 2  * * *
 3  * * 2a02:c7a:4:ffff::314 (2a02:c7a:4:ffff::314)  16.989 ms
 4  * * *
 5  2400:cb00:377:3:: (2400:cb00:377:3::)  19.047 ms  21.070 ms  20.285 ms
 6  2400:cb00:21:1024::8d65:6a1e (2400:cb00:21:1024::8d65:6a1e)
    18.522 ms  19.862 ms  17.842 ms

```

B.3.2 ‘*Intercontinental*’ RTT traces

To determine the ‘*intercontinental*’ glsrtt, glsrtt from a host in a domestic IPv6 network in St Andrews to hosts in other continents, such as America and Asia was used to determine the ‘*typical*’ glsrtt a host may encounter.

As shown, depending on the topological distance and physical distance, the RTT varies. ‘*Inter-continental*’ RTTs are beyond 100+ ms. Between the host in America and Asia, the ‘middle-point’ was $\tilde{200}$ ms. This is also the order of magnitude above the ‘national’ RTT.

The following shows the RTT measurements taken.

RTT to Hurricane Electric’s web server — America

```
ryo@Argus:~ $ ping6 ipv6.he.net
PING ipv6.he.net(ipv6.he.net (2001:470:0:64::2)) 56 data bytes
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=1 ttl=50 time=146 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=2 ttl=50 time=149 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=3 ttl=50 time=146 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=4 ttl=50 time=146 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=5 ttl=50 time=146 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=6 ttl=50 time=146 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=7 ttl=50 time=146 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=8 ttl=50 time=148 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=9 ttl=50 time=146 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=10 ttl=50 time=145 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=11 ttl=50 time=146 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=12 ttl=50 time=148 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=13 ttl=50 time=148 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=14 ttl=50 time=146 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=15 ttl=50 time=146 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=16 ttl=50 time=145 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=17 ttl=50 time=145 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=18 ttl=50 time=146 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=19 ttl=50 time=146 ms
64 bytes from ipv6.he.net (2001:470:0:64::2): icmp_seq=20 ttl=50 time=148 ms
^C
--- ipv6.he.net ping statistics ---
```

20 packets transmitted, 20 received, 0% packet loss, time 46ms
rtt min/avg/max/mdev = 145.422/146.305/149.145/1.045 ms

RTT to Open Computer Network (OCN) — Japan

```
ryo@Argus:~ $ ping6 www.ocn.ne.jp
PING www.ocn.ne.jp(2400:4040:5d:496:219:164:251:192
(2400:4040:5d:496:219:164:251:192)) 56 data bytes
64 bytes from 2400:4040:5d:496:219:164:251:192
(2400:4040:5d:496:219:164:251:192): icmp_seq=1 ttl=51 time=269 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
(2400:4040:5d:496:219:164:251:192): icmp_seq=2 ttl=51 time=277 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
(2400:4040:5d:496:219:164:251:192): icmp_seq=3 ttl=51 time=268 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
(2400:4040:5d:496:219:164:251:192): icmp_seq=4 ttl=51 time=269 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
(2400:4040:5d:496:219:164:251:192): icmp_seq=5 ttl=51 time=269 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
(2400:4040:5d:496:219:164:251:192): icmp_seq=6 ttl=51 time=269 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
(2400:4040:5d:496:219:164:251:192): icmp_seq=7 ttl=51 time=269 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
(2400:4040:5d:496:219:164:251:192): icmp_seq=8 ttl=51 time=269 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
(2400:4040:5d:496:219:164:251:192): icmp_seq=9 ttl=51 time=268 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
(2400:4040:5d:496:219:164:251:192): icmp_seq=10 ttl=51 time=268 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
(2400:4040:5d:496:219:164:251:192): icmp_seq=11 ttl=51 time=269 ms
```

```
64 bytes from 2400:4040:5d:496:219:164:251:192
  (2400:4040:5d:496:219:164:251:192): icmp_seq=12 ttl=51 time=277 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
  (2400:4040:5d:496:219:164:251:192): icmp_seq=13 ttl=51 time=269 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
  (2400:4040:5d:496:219:164:251:192): icmp_seq=14 ttl=51 time=269 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
  (2400:4040:5d:496:219:164:251:192): icmp_seq=15 ttl=51 time=269 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
  (2400:4040:5d:496:219:164:251:192): icmp_seq=16 ttl=51 time=268 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
  (2400:4040:5d:496:219:164:251:192): icmp_seq=17 ttl=51 time=268 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
  (2400:4040:5d:496:219:164:251:192): icmp_seq=18 ttl=51 time=269 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
  (2400:4040:5d:496:219:164:251:192): icmp_seq=19 ttl=51 time=272 ms
64 bytes from 2400:4040:5d:496:219:164:251:192
  (2400:4040:5d:496:219:164:251:192): icmp_seq=20 ttl=51 time=269 ms
^C
--- www.ocn.ne.jp ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 46ms
rtt min/avg/max/mdev = 268.342/269.630/277.417/2.679 ms
```

Appendix C

Multihoming implementation decisions

In the process of implementing multihoming with ILNPv6 in Linux kernel, several design decisions were made to retain the control over packet processing mechanism. This section of appendix discuss the circumstance within the Linux kernel and ILNPv6 source code that lead to the decisions made in implementing multihoming prototype.

In order to realise multihoming, a mechanism to distribute the packets to available paths is required. Such mechanism is often referred to as a scheduler. In Linux kernel, there are several schedulers implemented within `net/sched` as `sched_*.c` files. (e.g. `net/shed/shed_drr.c`—Deficit Round Robin scheduler) They are the packet processing schedulers configurable from the `tc` program (traffic control) to achieve either some sort of traffic shaping or emulation. While there are several methods to realise 'multihoming' using `tc`, it involves an abstract interface with dedicated `qdisc` to utilise two links between two specific host—A mechanism that involves disabling source address validation as the packets are addressed and sent from an abstract interface with separate ip addresses to physical interfaces, which will have to reside in the same subnet. The schedulers in `net/sched` takes `struct Qdisc` and `struct sk_buff` to enqueue, and dequeue to process the packet. While the algorithm may be useful in load balancing across different L64, the process of adapting the mechanism to utilise them appear not trivial at all. It

may need once again another abstract interface to attach qdisc and scheduler to, without L64 then to the physical interfaces with L64. However, ILNPv6—to achieve clean soft-handoff that reflects the state of L64 accurately—already have taken control of route, and output interface selection at the transport packet processing path. Due to the nature of current 'IP address' use in the transport, and network layers as well as the statically bound physical interface to the IP address, the kernel decides the specific output interface and IP address in the transport layer packet processing paths. Such process is known as '*corking*'. Current ILNPv6 mobility directly modifies the output interface and '*IPv6 subnet*' — L64 in ILNPv6 context — in the '*corking*' phase.

Since mobility mechanism introduced in this project is already intervening the corking process, it is only natural that the mobility mechanism directly utilise the facility in place to add a scheduler/load-balancer. Especially due to the fact that this is a proof-of-concept and is not intended as the permanent solution.

Appendix D

Behaviours observed in the evaluation testbed

D.1 Interaction between TCP window scaling, iperf, and throughput visualisation in high-latency links

Transmission Control Protocol (TCP) has a mechanism to control the flow and to guarantee that the segment is correctly sent and received in order known as sliding window, or just TCP window. One of the common performance optimisation implemented in the protocol is to increase its size of the window at a given scale. In order to scale the window, several rounds of packets has to be exchanged before reaching a stable state of the window. As the round trip time (RTT) increases, the time duration required to reach a stable window state increases. During the time the TCP window is scaling up, the upper layer applications will continue to push data to the socket API. In this case, `iperf` will continue to push the data at a set rate configured through the command-line argument. However, until the stable state of TCP window is reached, the packet will be sent as bursts. Every time an acknowledgement packet comes through at each one RTT timing, TCP will fill the window by sending the packets in the queue as the acknowledgement pushes the window forward. If the nodes are communicating over the RTT link, the gap between each bursts

that fills the TCP window grows, causing larger queue. In the initial part of this phase, the bursts may happen at a rate below the capacity of the link, however, as the window scales, the bursts will happen at a larger chunks, with higher peaks of throughput. Since none of the behaviours are visible from the upper layer applications using the socket API, if an application is attempting to pace the data throughput, the actual throughput will not match what the application is aiming to achieve. In this specific context, while `iperf` may continue to push packets at, for example, 10 Mbps, each occurrences of the burst may begin lower than that, then potentially exceed before reaching the desired throughput. If the link itself is capable of carrying such bursts beyond the set throughput, and this phase of window scaling takes a significant amount of time, it will result in a larger spike well beyond the set throughput as more packets accumulate in the queue, while the sender is waiting for the acknowledgement packet to be able to advance the window and grow the window.

In this study, we have observed a significant bursts when 200ms delay was applied. In the time series throughput diagram, such effect can be amplified depending on the visualisation method. In this analysis, I have opted to calculate the throughput by aggregating bytes received at a set duration, then taking the division by the aggregation duration. This was set to 0.5 seconds, which is beyond the RTT leading to contain multiple sets of bursts.

Appendix E

Real time protocol (RTP) experiment and video metrics

E.1 VQMT analysis

VQMT tools provided SSIM and PSNR analysis. The following describes how VQMT was used in the data analysis process.

In order for VQMT to carry out frame-by-frame analyses, the video clips have to be re-encoded into raw YUV format. However, the video was transmitted in H.264 MPEG-TS format. Therefore, ffmpeg was used to re-encode and format them in raw YUV file. A bash script was used to automate the video re-encoding and analysis processes. The following shows an excerpt of the script relevant to re-encoding the video

```
1 pix_fmt="yuv420p"
2 ffmpeg_args="-err_detect ignore_err -i $source_file -pix_fmt "$pix_fmt" \
3     -c:v rawvideo -map 0:v -f rawvideo $output_filename"
4
5 ffmpeg_args="-y $ffmpeg_args"
6
7 ffmpeg $ffmpeg_args
```

Once both the captured video clip and the reference source clip are re-encoded to raw YUV files, VQMT program is called to begin the frame-by-frame analysis. VQMT requires following parameters:

1. input raw yuv file path
2. “processed” capture raw yuv file path
3. output path
4. width
5. height
6. frame count
7. Chroma format
8. metrics to calculate (such as SSIM and PSNR)

Chroma format is an integer option with following presets:

- 0: YUV400 — YUV 4:0:0 subsampling
- 1: YUV420 — YUV 4:2:0 subsampling
- 2: YUV422 — YUV 4:2:2 subsampling
- 3: YUV444 — YUV 4:4:4 subsampling

The 4:2:0 chroma subsampling is used with MPEG videos, therefore the chroma format option was set to 1. ffprobe was used to measure and verify the video attributes such as its dimension and frame count. Once they are gathered, VQMT program is called. The following shows how the program is called in the script:

```
1 # build vqmt arguments
2 vqmt_args="$reference_yuv_path $captured_yuv_path $width $height \
3           $frame_count $colour_profile $output_filename \
4           $values_to_calculate"
```

5

```
6 vqmt $vqmt_args || echo "Error with vqmt"
```

E.2 Acquiring and preparation of the video samples

The original video files were acquired from the respective URLs as shown in Table 7.1. For each resolution configurations, ffmpeg was used to trim the video down to 120 seconds clips. BBB clips began at the start of the video, while ToS clips began later, from 6 minutes. The following command was used to trim ToS clips:

```
ffmpeg -ss 6:00.00 -start_at_zero -t 2:00.00 -i tears_of_steel_720p.mov \
-c:v libx264 -x264opts force_cfr=1 -framerate 24.000 -asinc 1 \
-c:a mp3 Tears_of_Steel-720p-h264-120s-encoded.mp4
```

The following command was used to trim BBB clips:

```
ffmpeg -ss 0:00.00 -i big_buck_bunny_720p_h264.mov -start_at_zero -t 2:00.00 \
-c:v libx264 -x264opts force_cfr=1 -framerate 24.000 -c:a mp3 -b:a 192k \
-async 1 big_buck_bunny-720p-h264-120s-explicit-start_zero.ts
```

The above commands will generate MPEG-TS clips with H.264 encoding. Those MPEG-TS clips were used as payload, transmitted using ffmpeg, and received by vlc.

Appendix F

Enlarged plots

This appendix contains the enlarged typical run plots for a reference.

F.1 Continuous mobility typical runs

The following shows the enlarged plots from continuous mobility typical runs as shown in Chapter 5, Section 5.4.1.

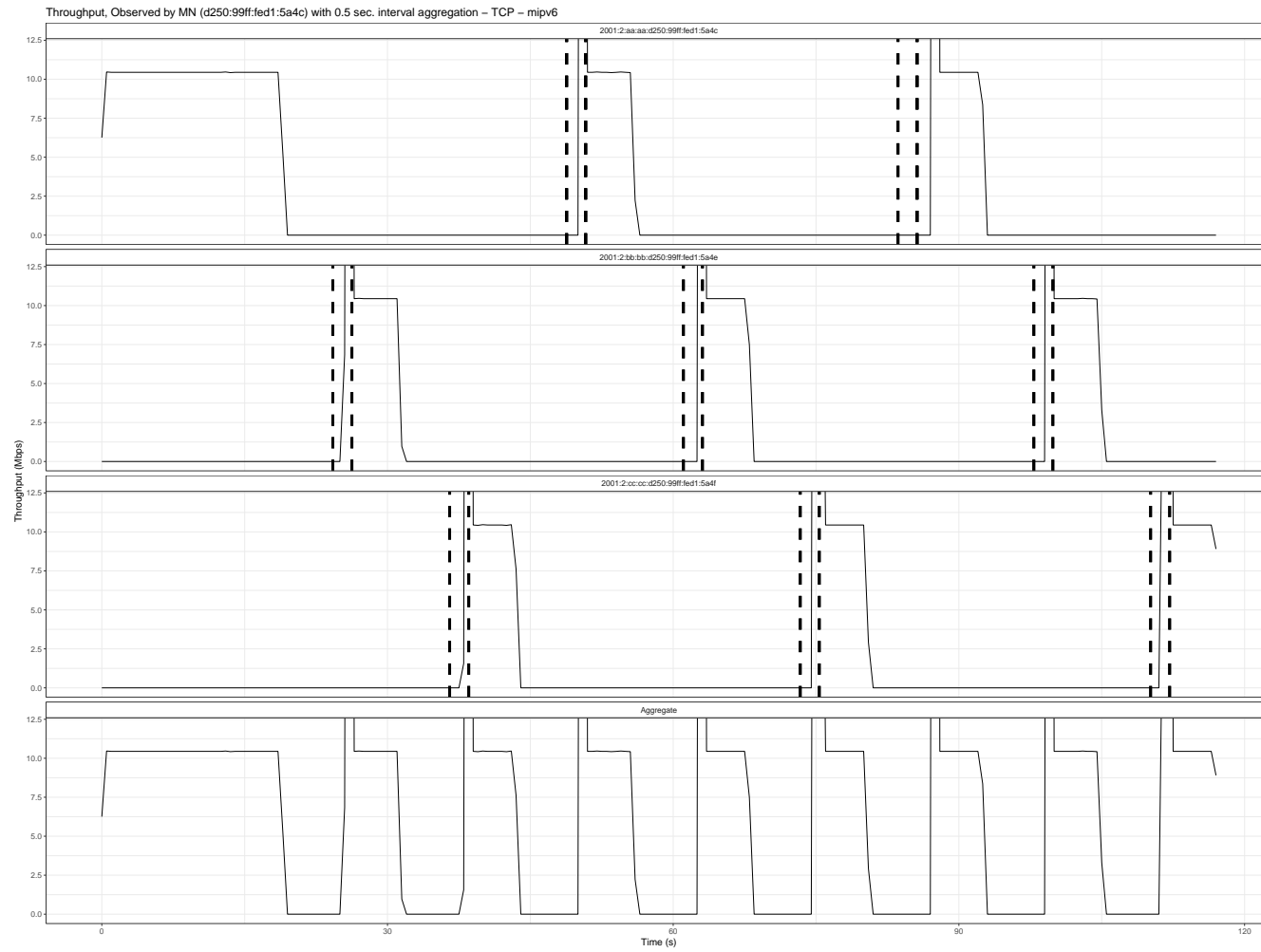


Figure F.1: Enlarged plots of continuous mobility MIPv6 TCP typical run throughput.

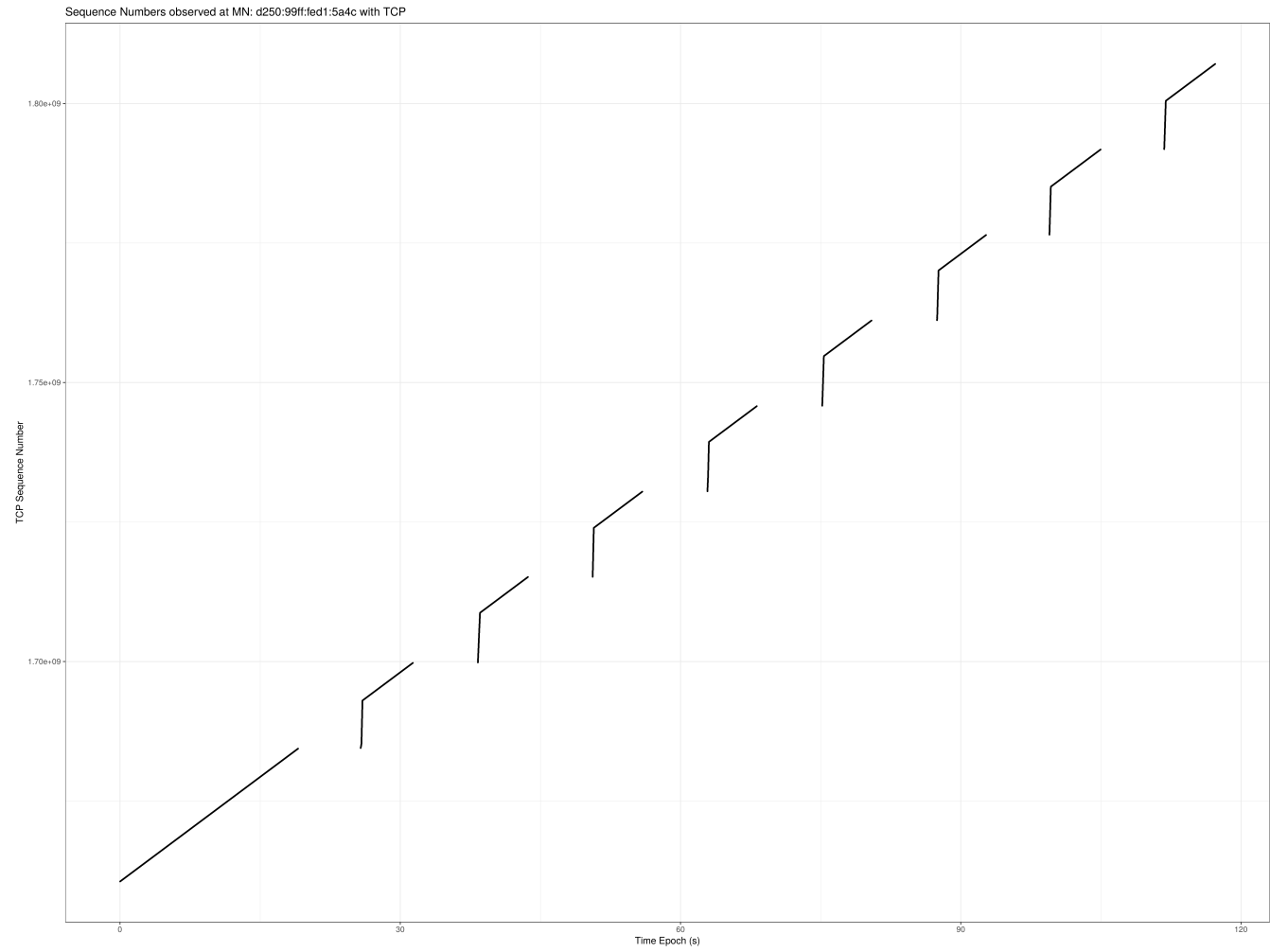


Figure F.2: Enlarged plot of continuous mobility MIPv6 TCP typical run sequence numbers.

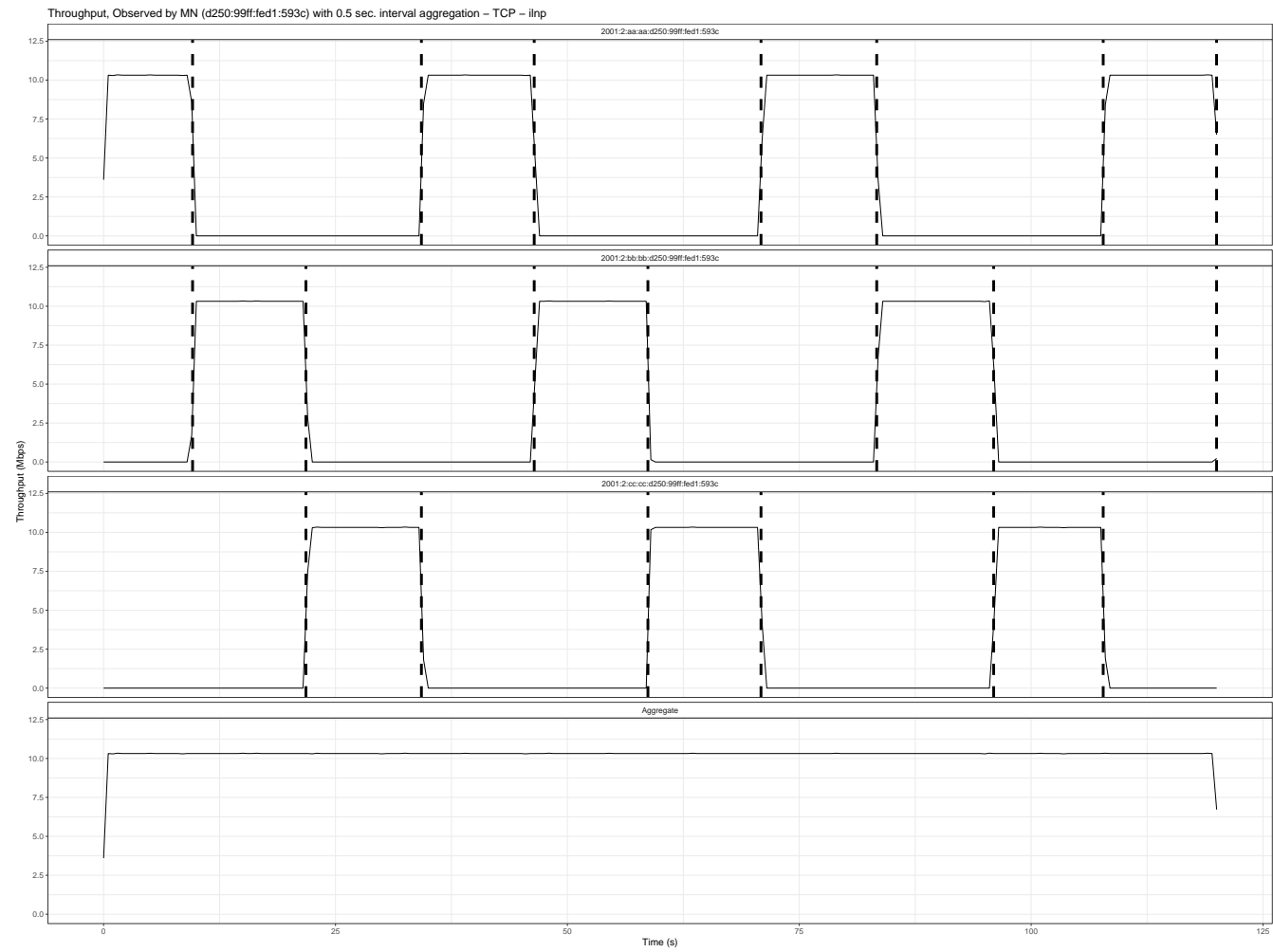


Figure F.3: Enlarged plots of continuous mobility ILNPv6 TCP typical run throughput.

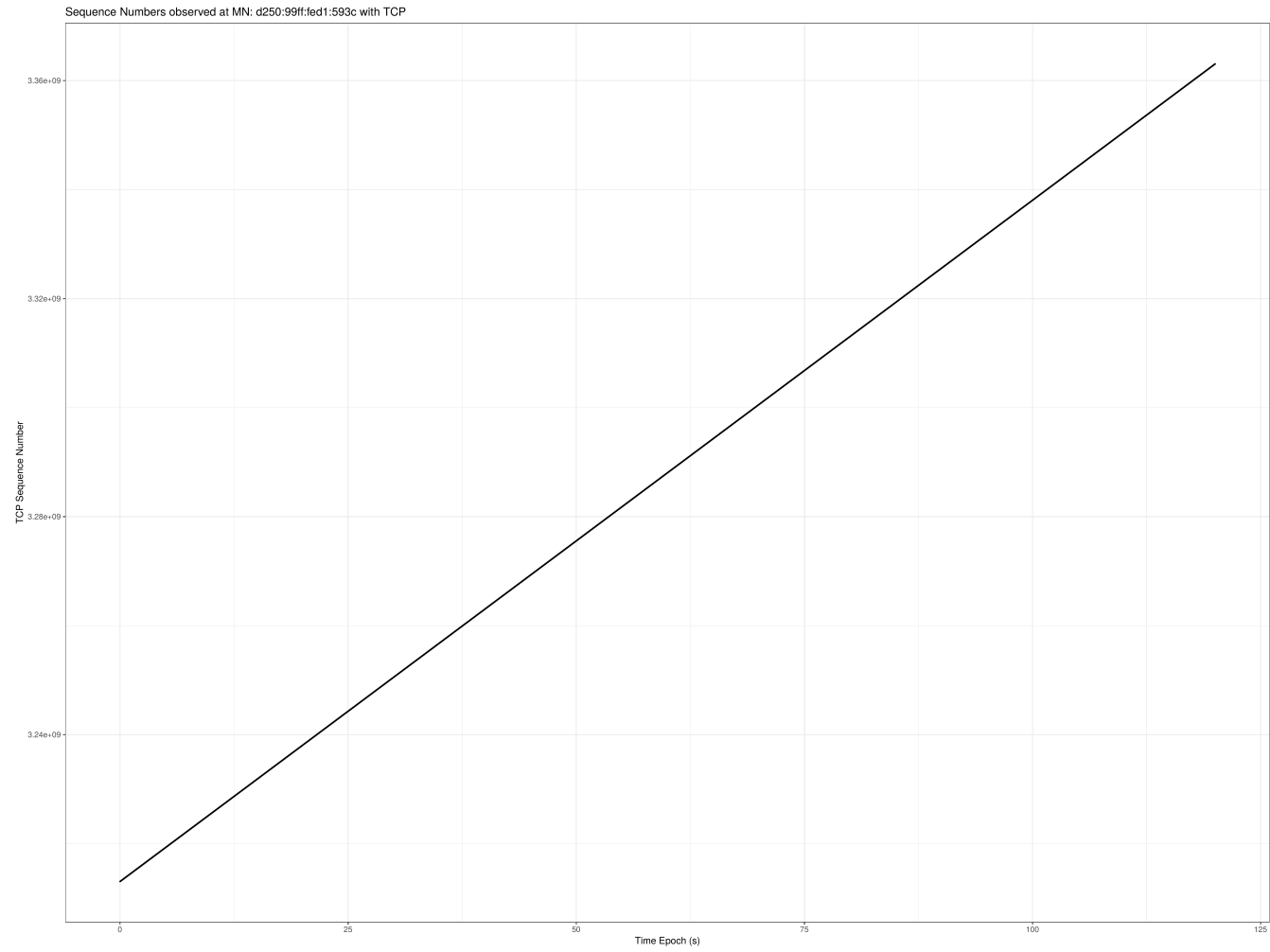


Figure F.4: Enlarged plot of continuous mobility ILNPv6 TCP typical run sequence numbers.

F.2 Mobility-multihoming duality typical runs

The following shows the enlarged plots from mobility-multihoming duality typical runs as shown in Chapter 6, Section 6.4.1.

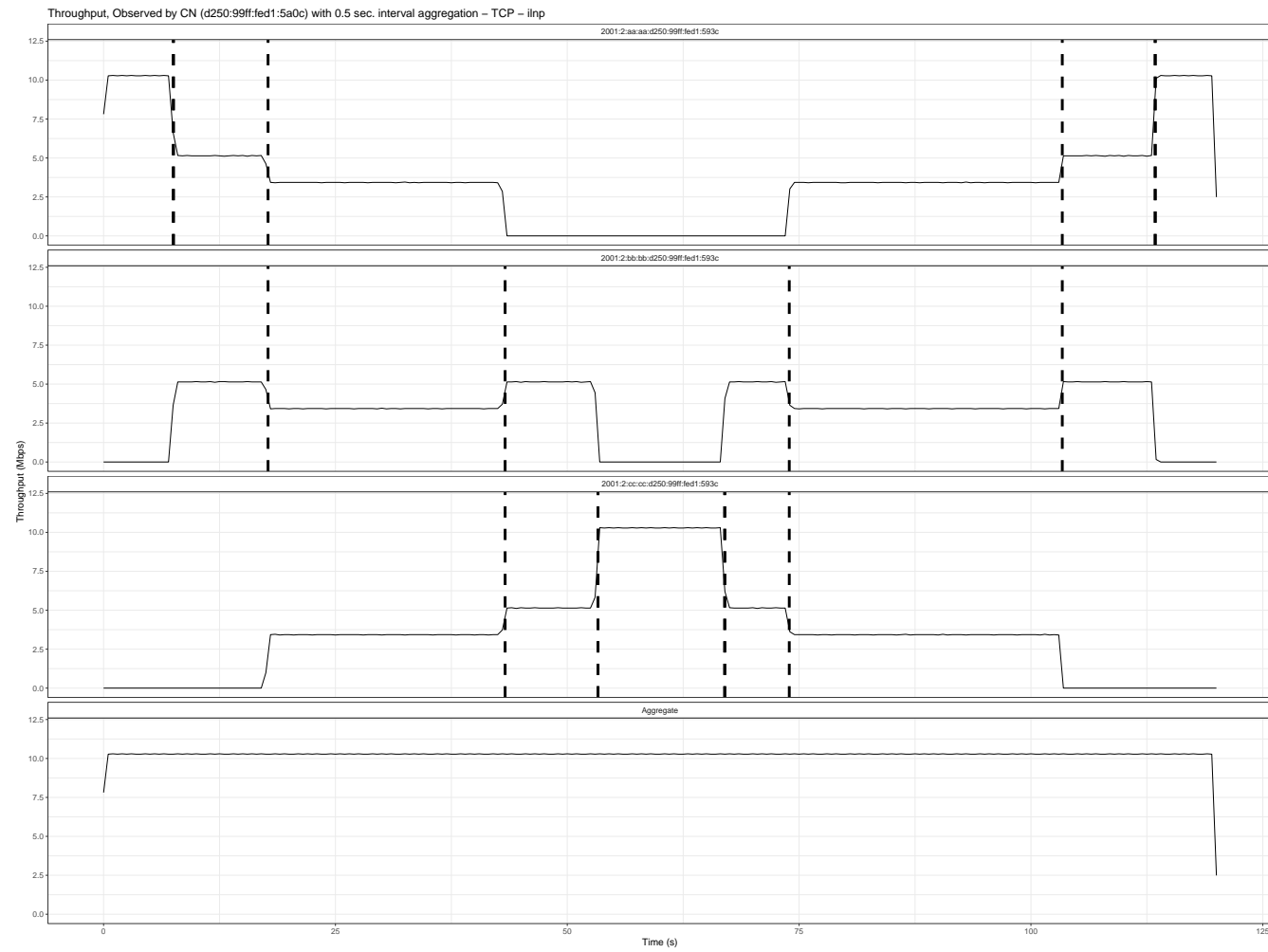


Figure F.5: Enlarged plots of mobility-multihoming duality ILNPv6 TCP typical run throughput at the CN.

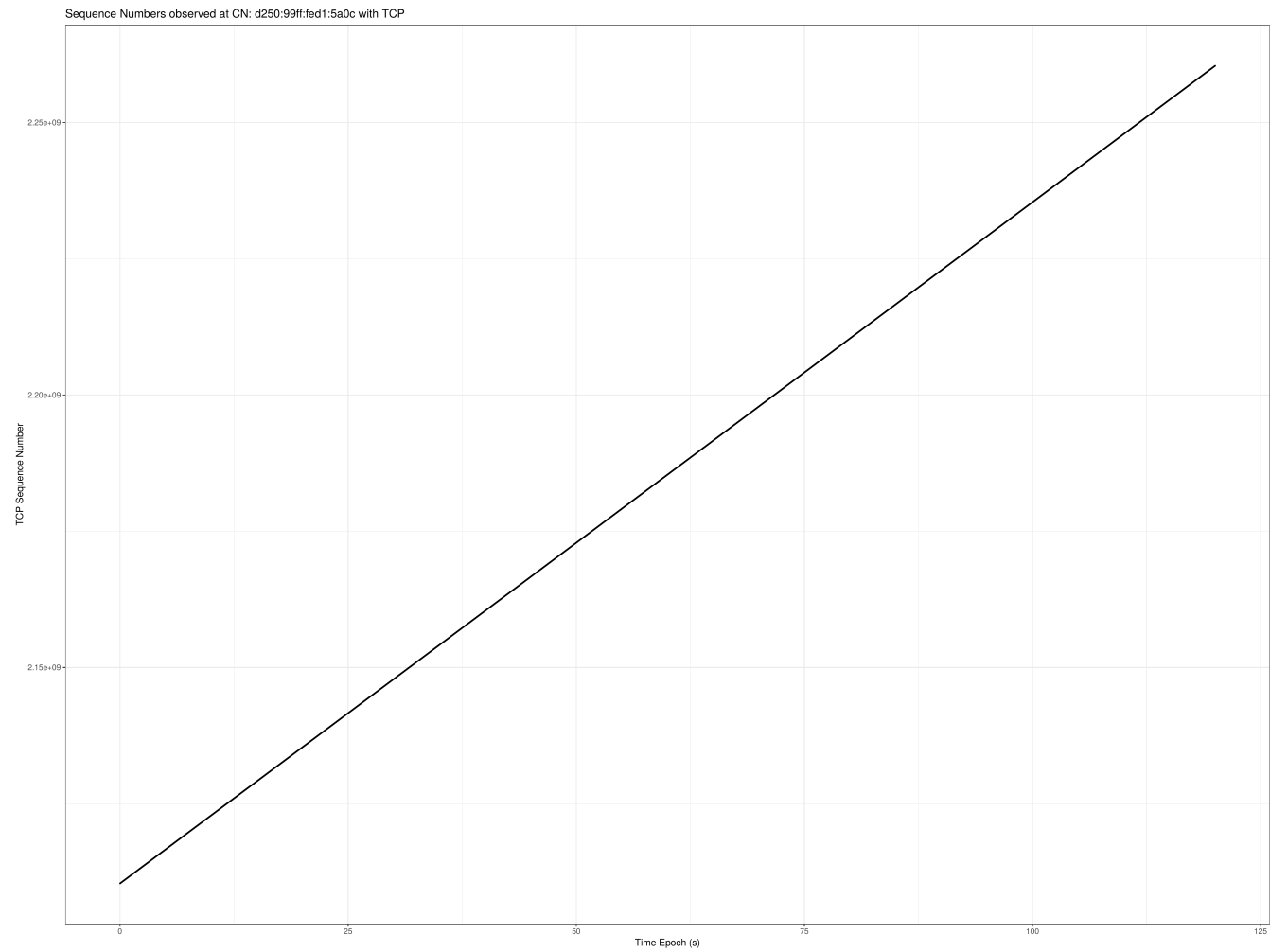


Figure F.6: Enlarged plot of mobility-multihoming duality ILNPv6 TCP typical run sequence numbers at the CN.

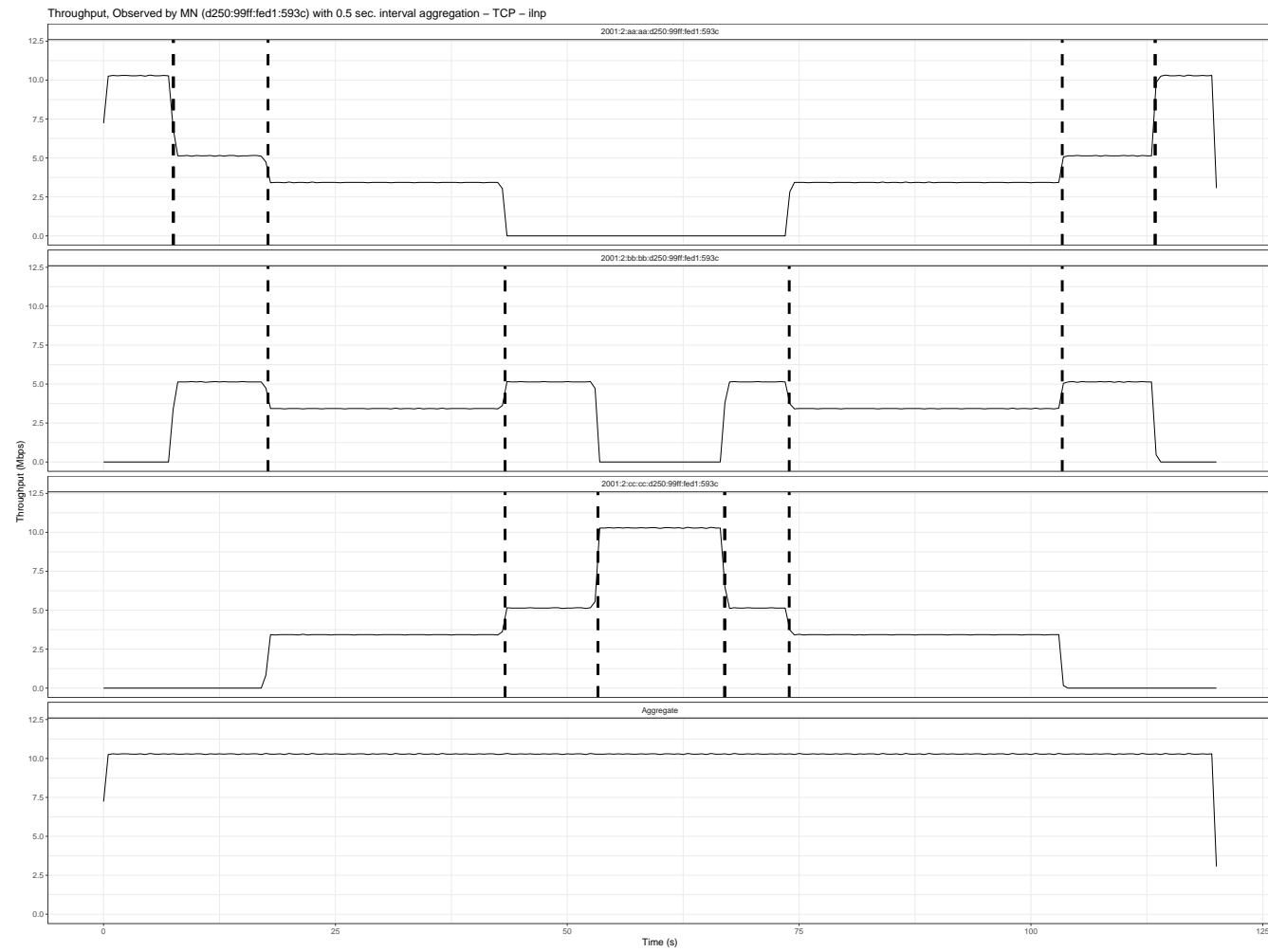


Figure F.7: Enlarged plots of mobility-multihoming duality ILNPv6 TCP typical run throughput at the MN.

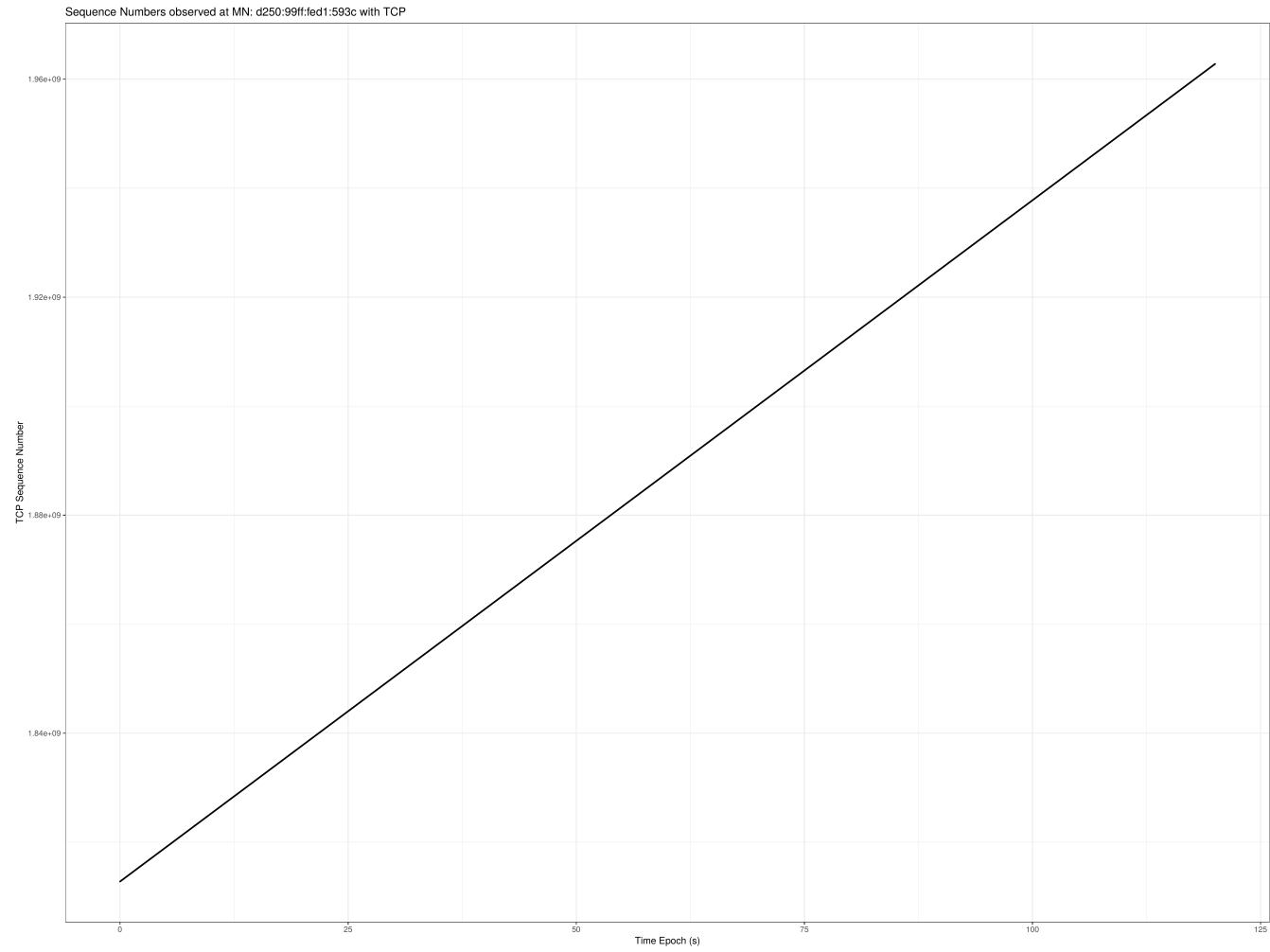


Figure F.8: Enlarged plot of mobility-multihoming duality ILNPv6 TCP typical run sequence numbers at the MN.

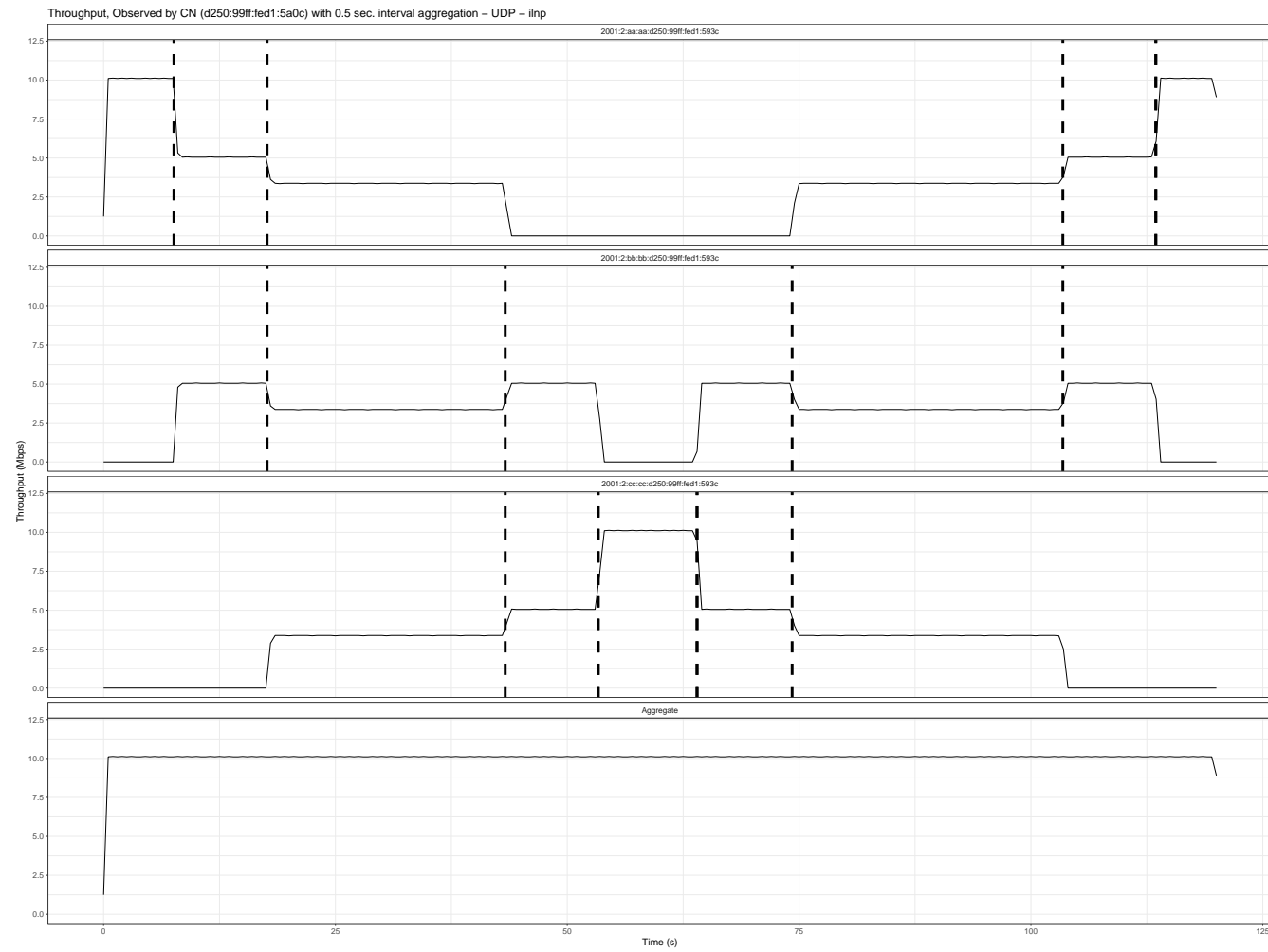


Figure F.9: Enlarged plots of mobility-multihoming duality ILNPv6 UDP typical run throughput at the CN.

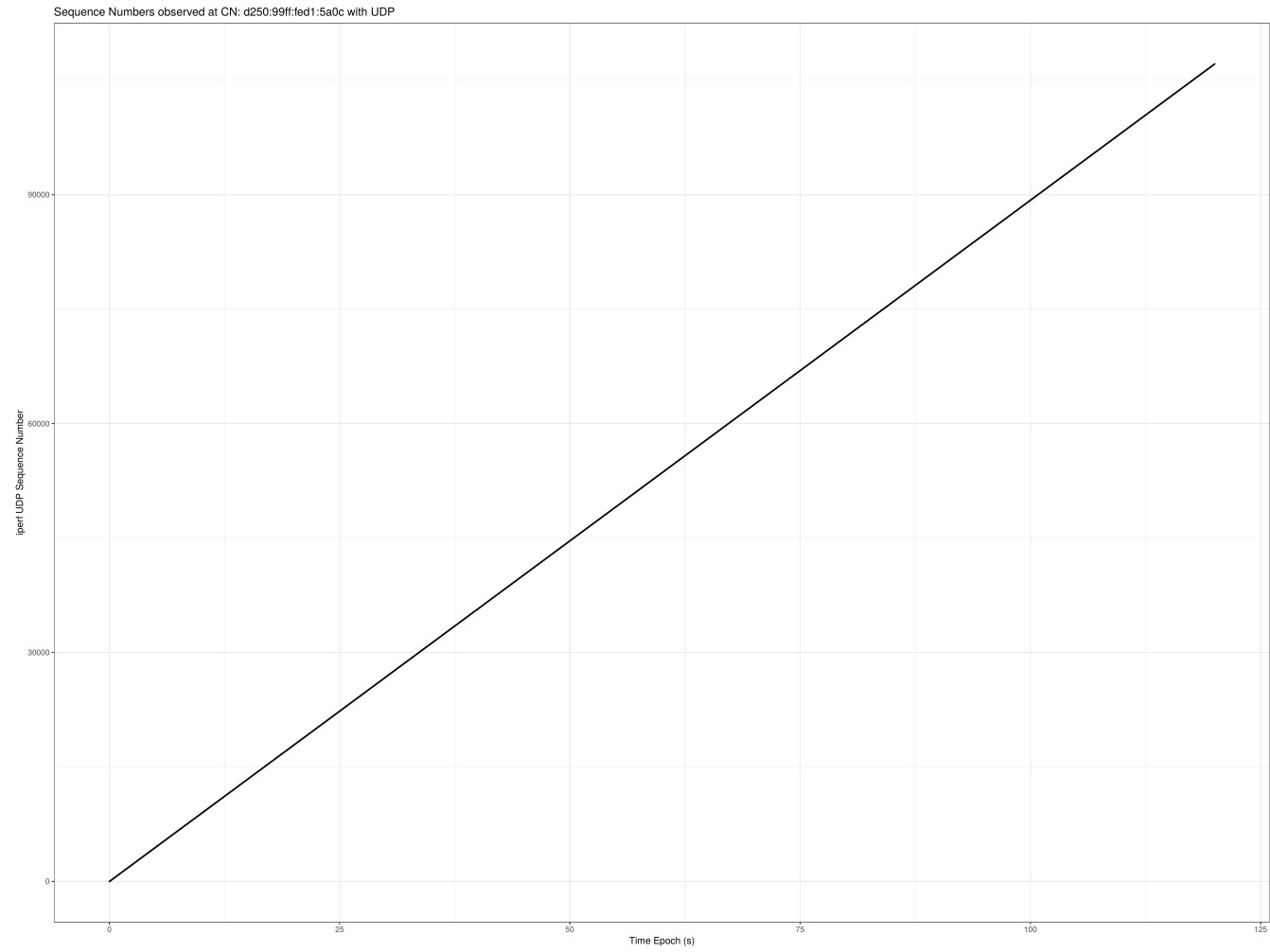


Figure F.10: Enlarged plot of mobility-multihoming duality ILNPv6 UDP typical run sequence numbers at the CN.

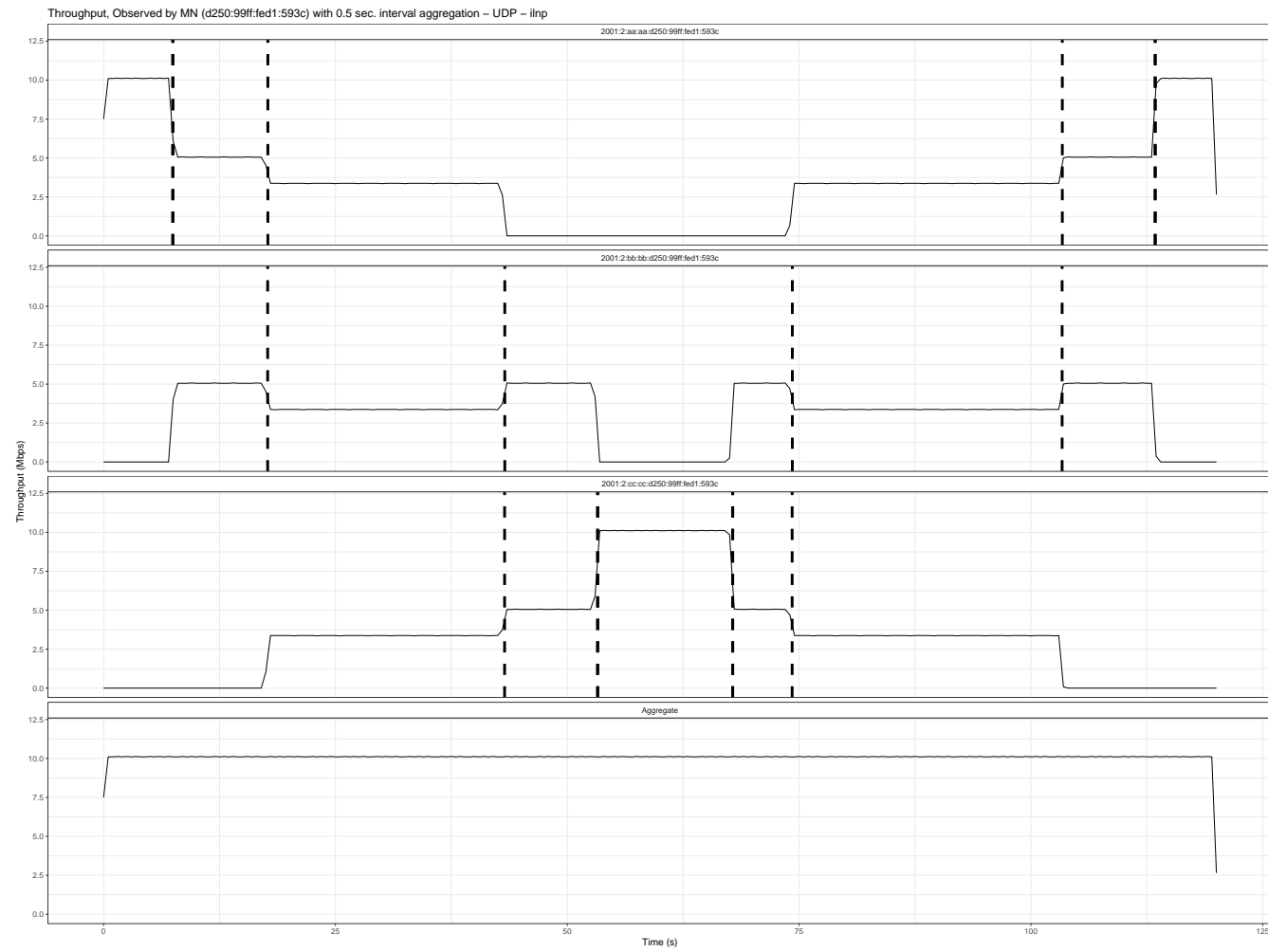


Figure F.11: Enlarged plots of mobility-multihoming duality ILNPv6 UDP typical run throughput at the MN.

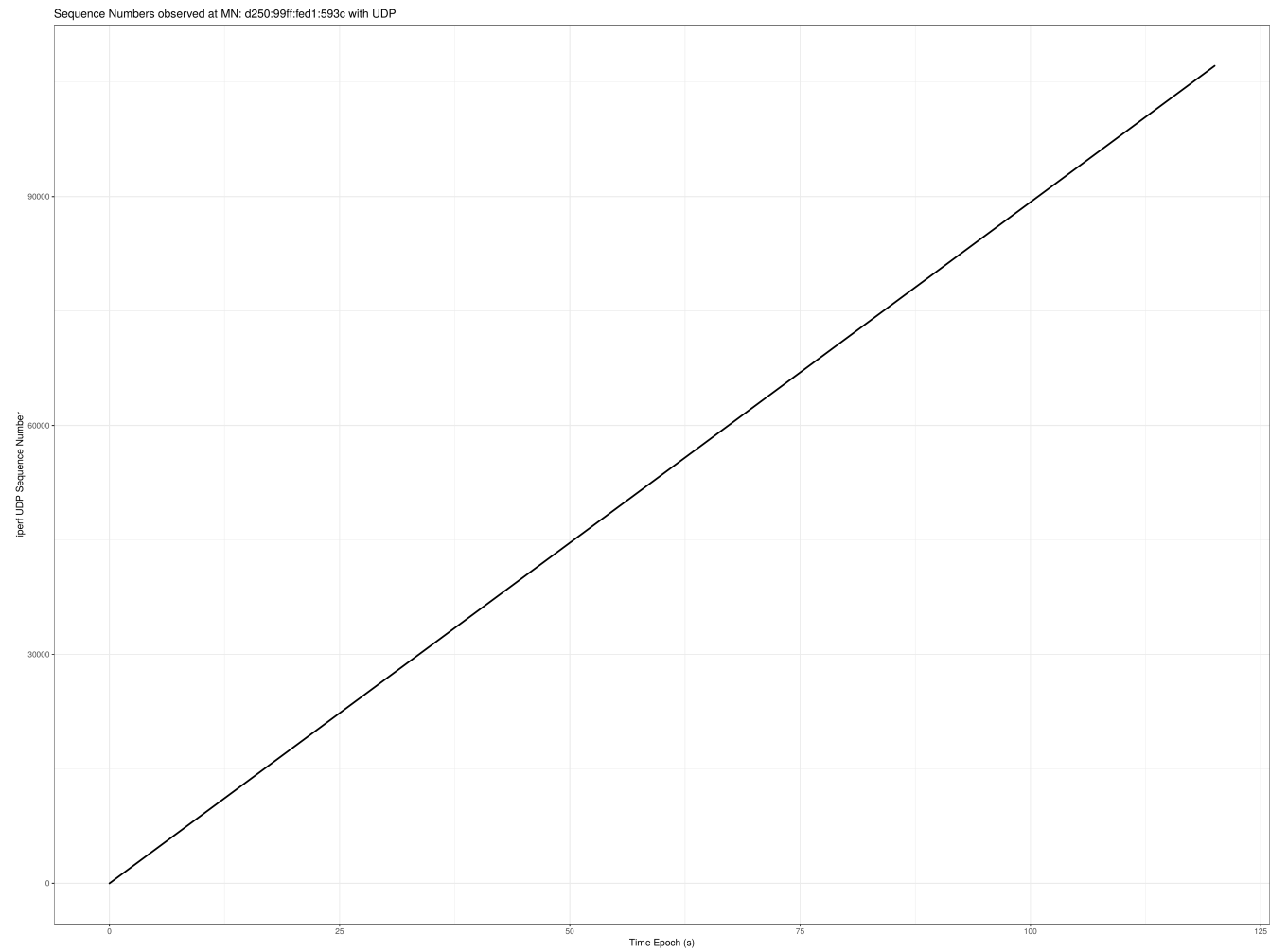


Figure F.12: Enlarged mobility-multihoming duality ILNPv6 UDP typical run sequence numbers at the MN.

F.3 Real-time video continuous mobility typical runs

The following shows the enlarged plots from real-time video continuous mobility scenario typical runs shown in Chapter 7, Section 7.5.2.

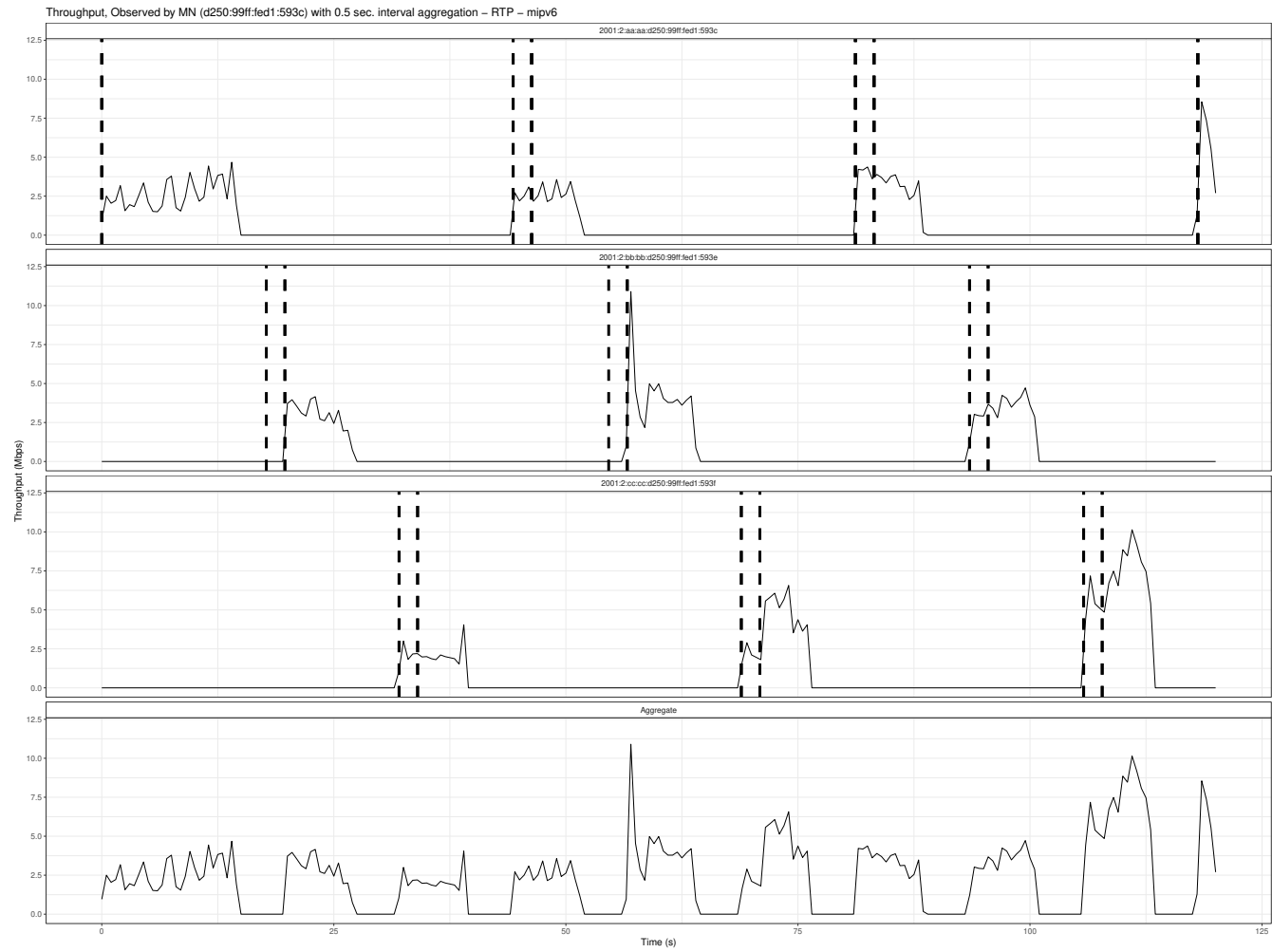


Figure F.13: Enlarged plots of mobility-multihoming duality MIPv6 RTP typical run throughput for ToS.

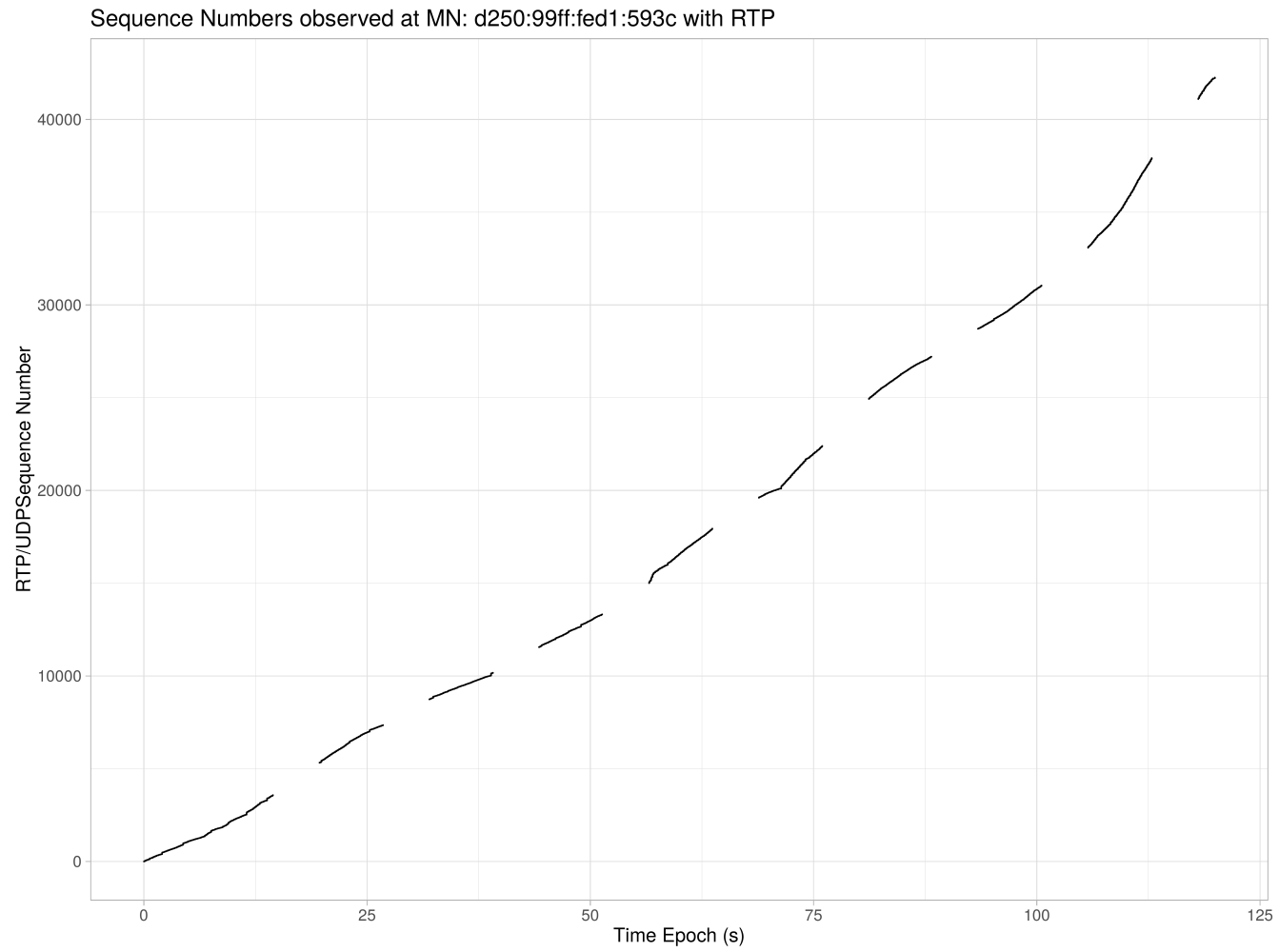


Figure F.14: Enlarged plot of mobility-multihoming duality MIPv6 RTP typical run sequence numbers for ToS.

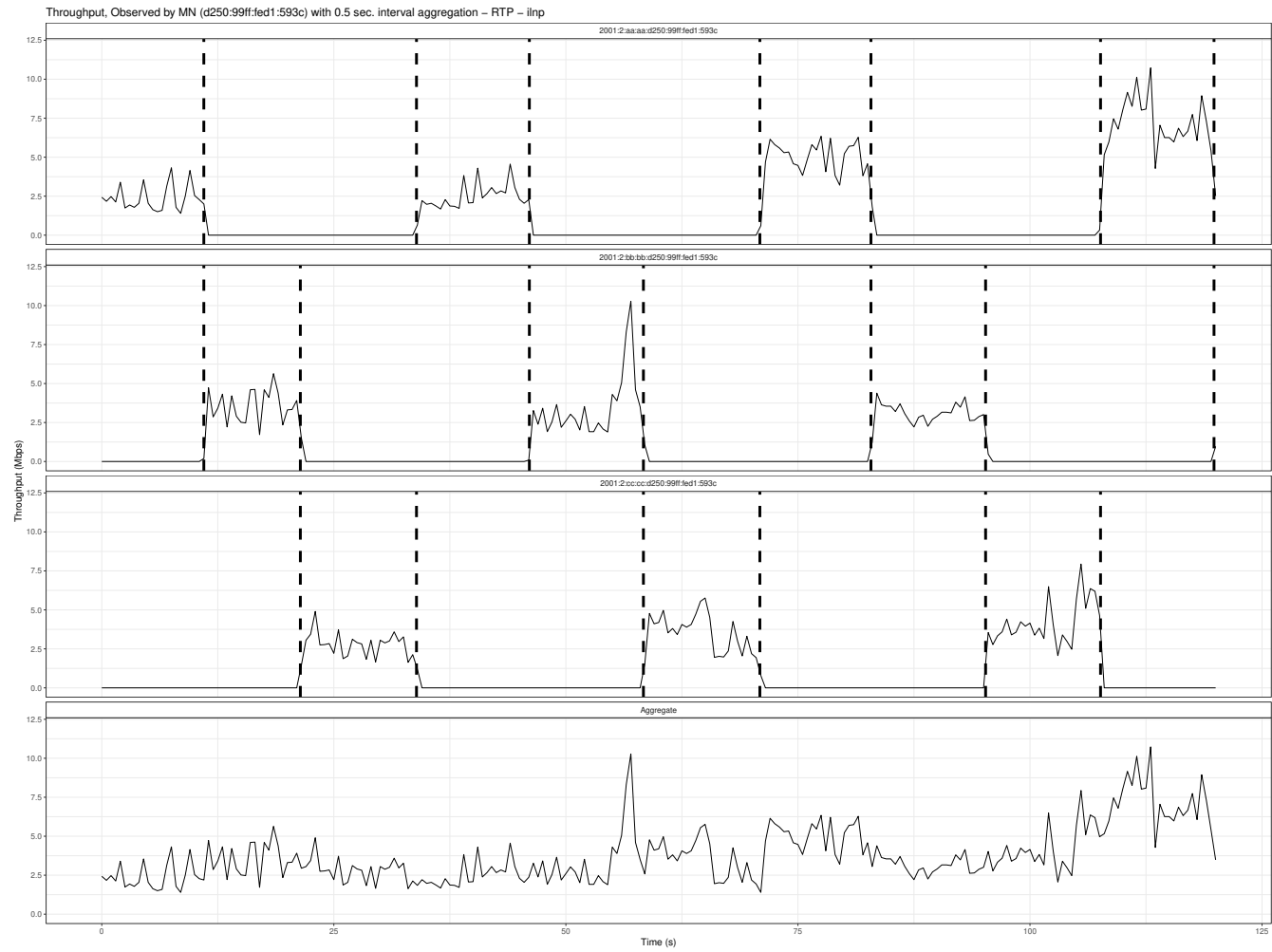


Figure F.15: Enlarged plots of mobility-multihoming duality ILNPv6 RTP typical run throughput for ToS.

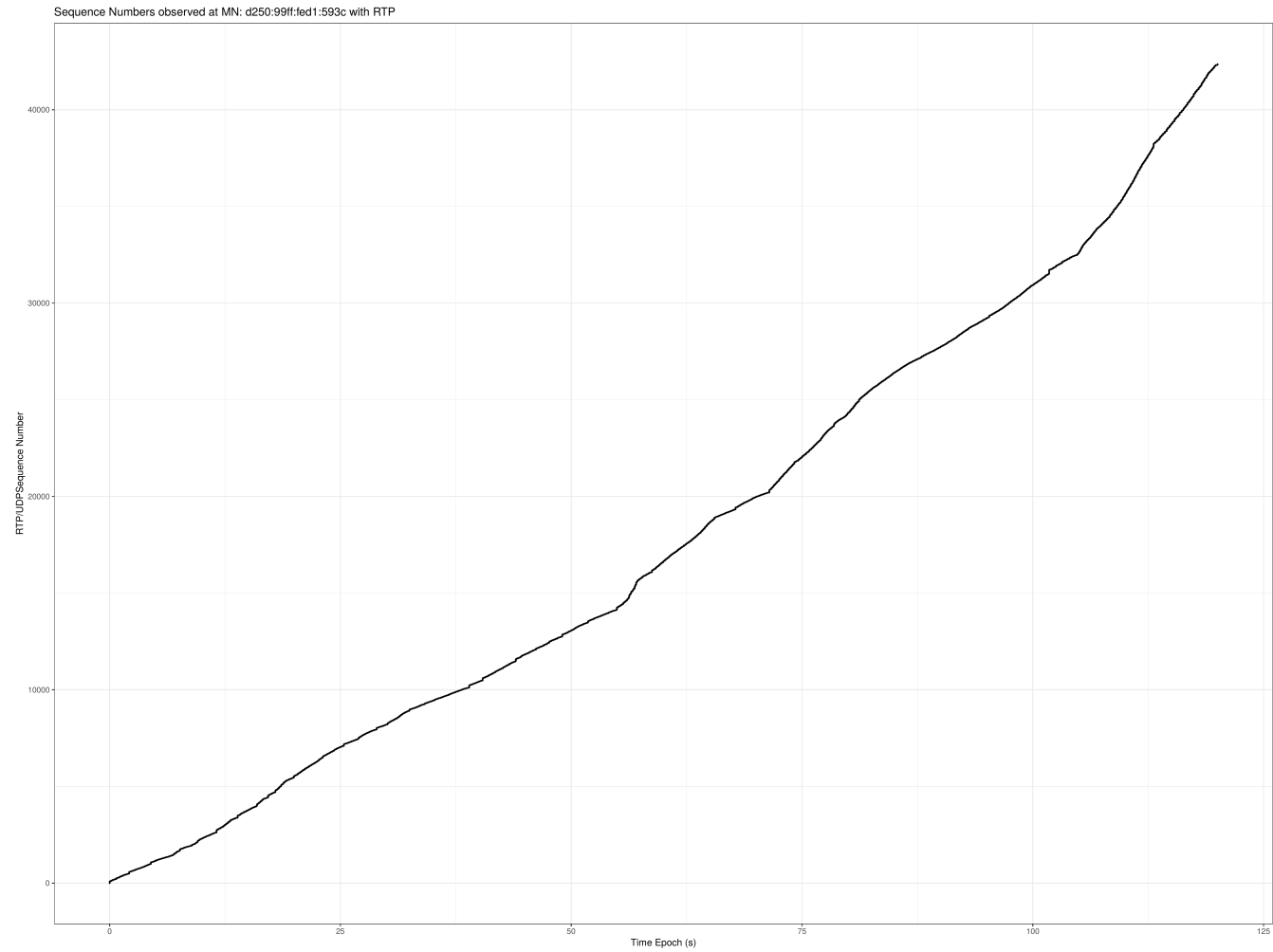


Figure F.16: Enlarged plot of mobility-multihoming duality ILNPv6 RTP typical run sequence numbers for ToS.

F.4 Real-time video mobility-multihoming duality runs

The following shows the enlarged plots from real-time video mobility-multihoming duality typical runs shown in Chapter 7, Section 7.6.2.

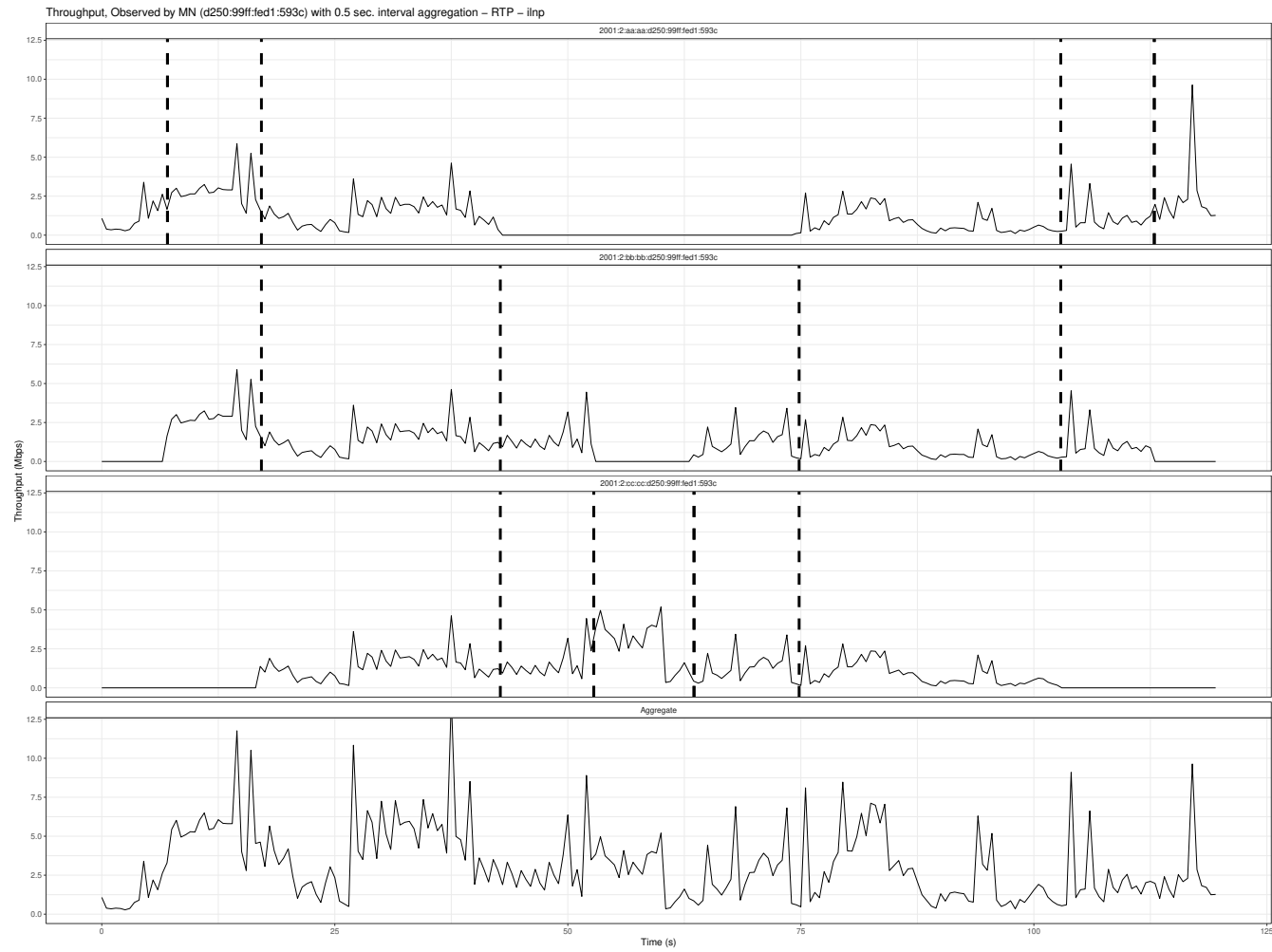


Figure F.17: Enlarged plots of continuous mobility ILNPv6 RTP typical run throughput for BBB.

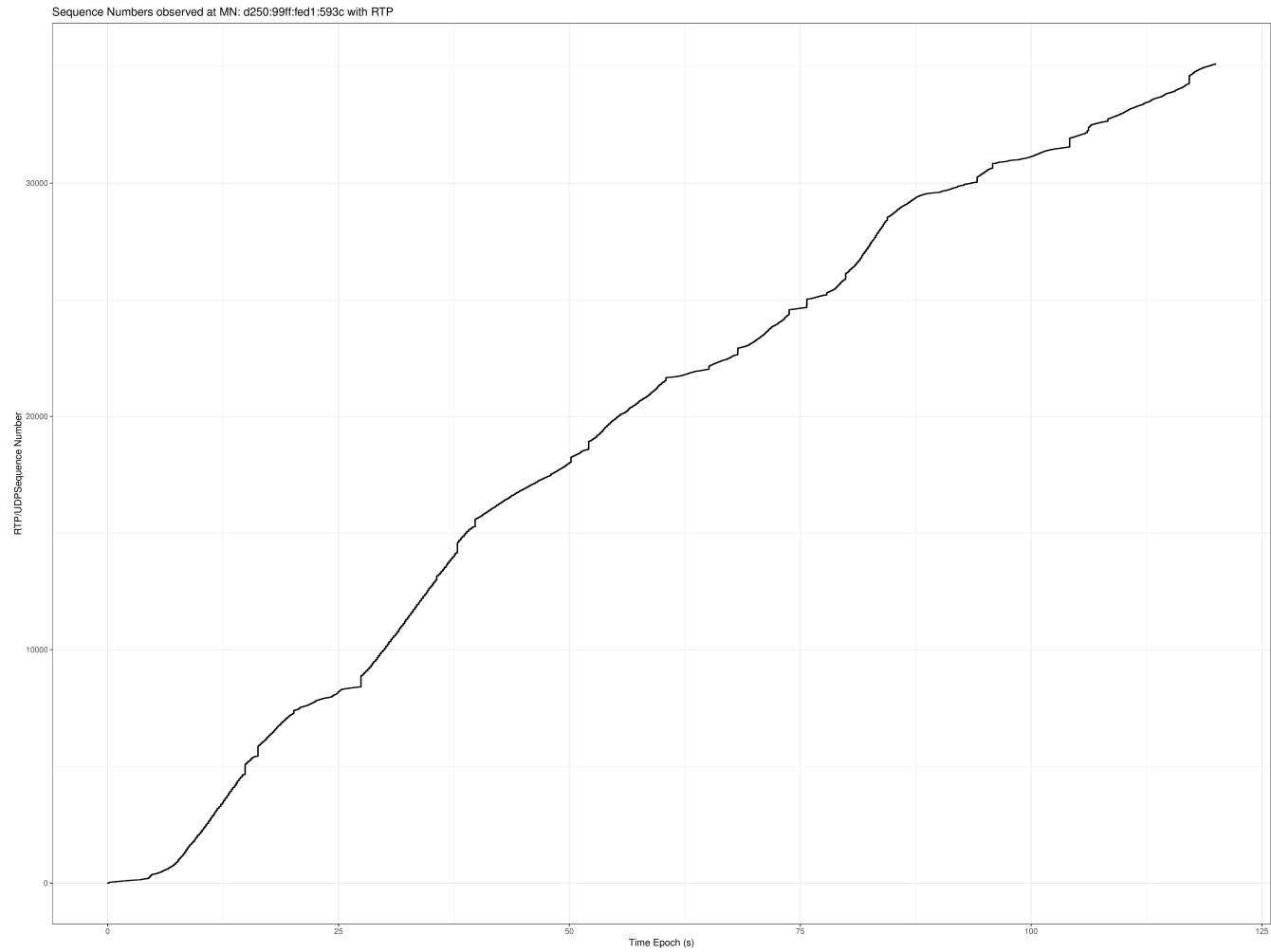


Figure F.18: Enlarged plot of mobility-multihoming duality ILNPv6 UDP typical run sequence numbers for BBB.

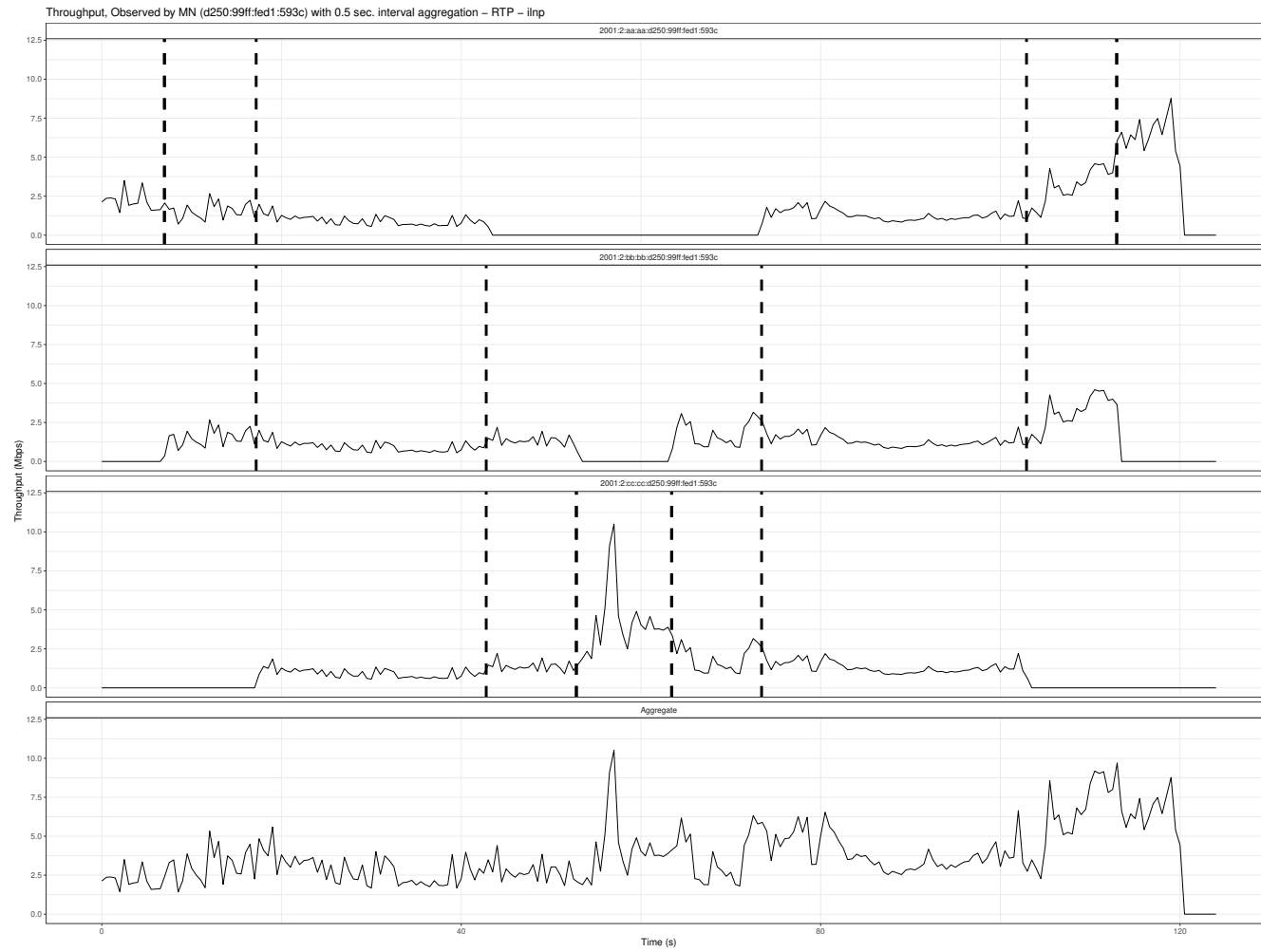


Figure F.19: Enlarged plots of continuous mobility ILNPv6 RTP typical run throughput for ToS.

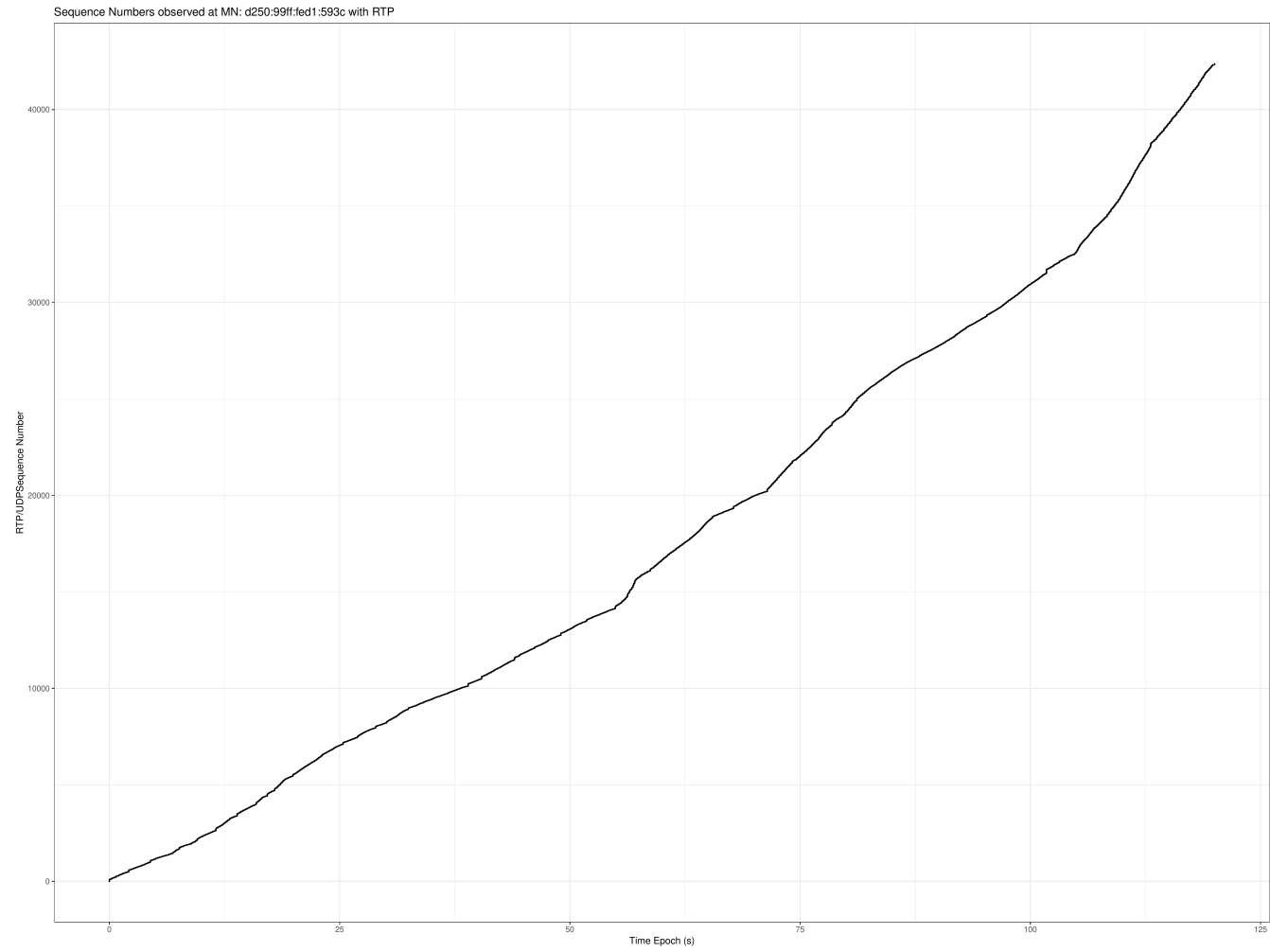


Figure F.20: Enlarged plot of mobility-multihoming duality ILNPv6 UDP typical run sequence numbers for ToS.

F.5 MP-TCP example runs

The following shows the enlarged plots from Chapter 2, used in qualitative analysis of MP-TCP behaviour.

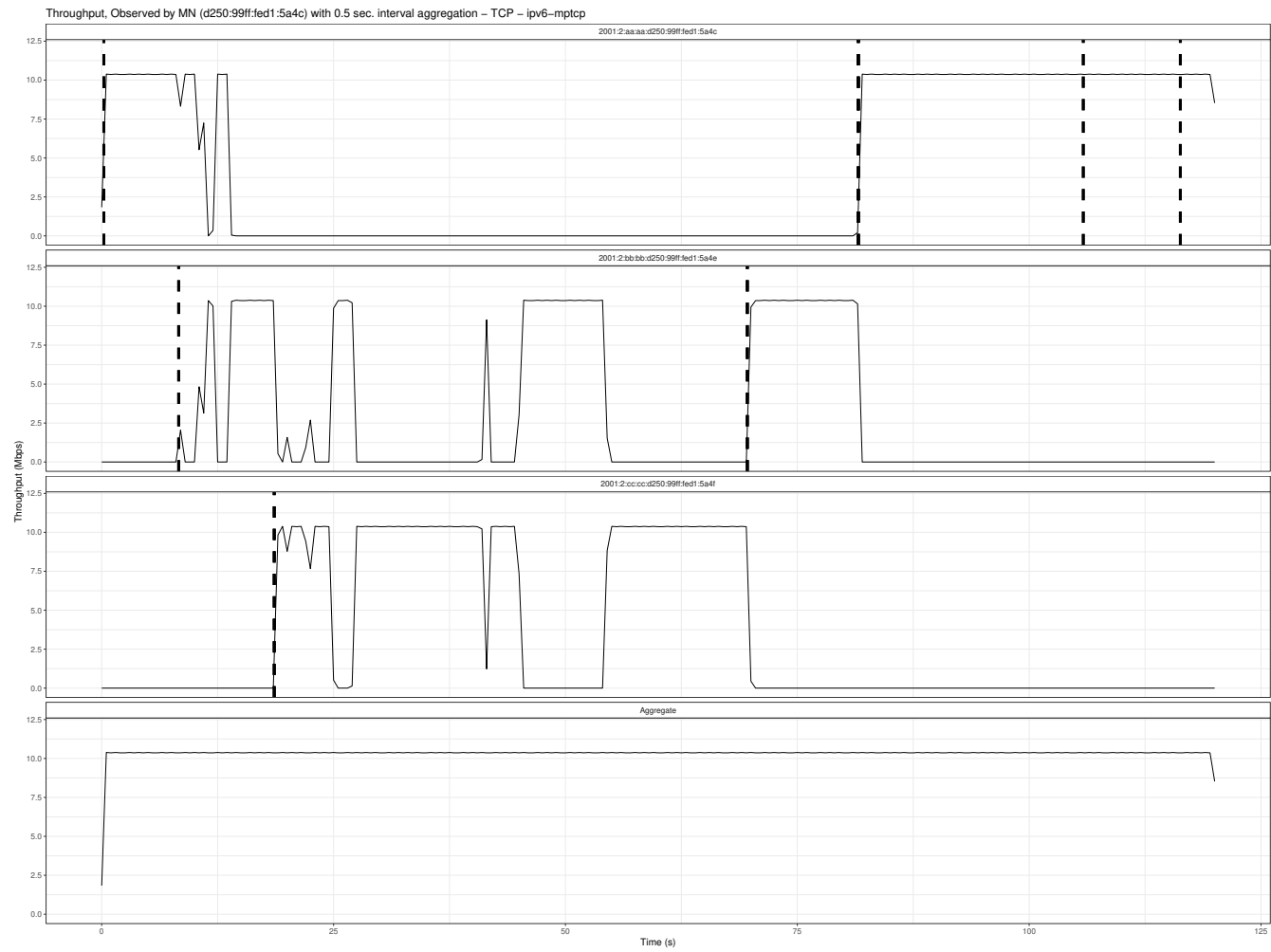


Figure F.21: Enlarged MP-TCP throughput facet graph at the MN.

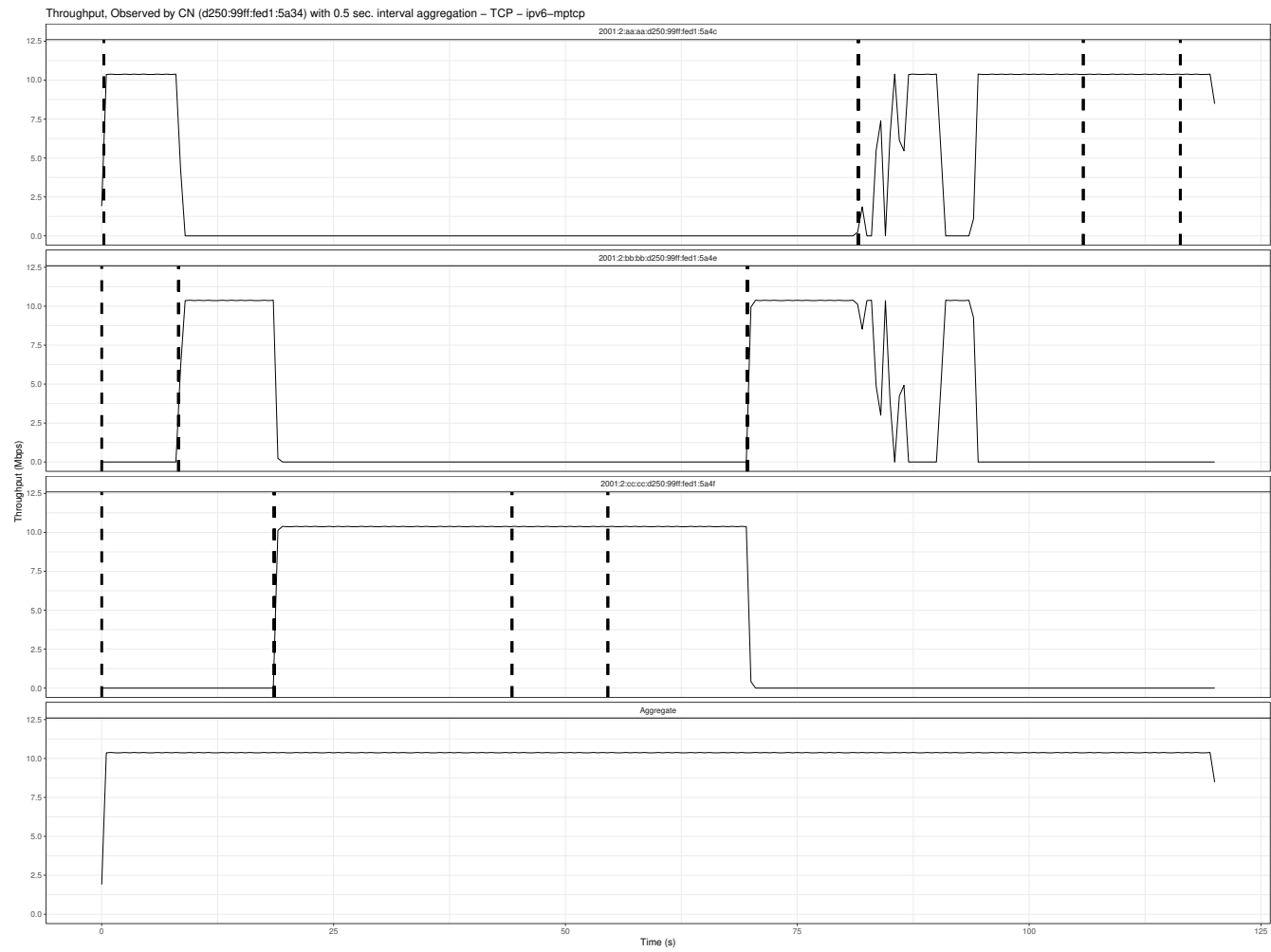


Figure F.22: Enlarged MP-TCP throughput facet graph at the CN.

Appendix G

Hardware used in evaluation

For the purpose of reproducibility, the following contains excerpts from the datasheet of the equipment used.

G.1 Network equipment

The evaluation testbed consisted of following two models of routers from Ubiquiti Networks. The following are the excerpts of datasheets published by the manufacturer.

Figure G.1: Datasheet of EdgeRouter X from Ubiquiti Networks website: https://dl.ubnt.com/datasheets/edgemax/EdgeRouter_X_DS.pdf

G.1.1 Ubiquiti Networks EdgeRouter X

ER-X	
Dimensions	110 x 75 x 22 mm (4.33 x 2.95 x 0.87")
Weight	175 g (6.17 oz)
Max. Power Consumption	5W
Power Input	12VDC, 0.5A Power Adapter (Included) or 24V Passive PoE
Power Supply	External AC/DC Adapter
Supported Voltage Range	9 to 26VDC
Button	Reset
LED	Power, Ethernet 0-4
Processor	Dual-Core 880 MHz, MIPS1004Kc
System Memory	256 MB DDR3 RAM
Code Storage	256 MB NAND
Certifications	CE, FCC, IC
Wall-Mount	Yes
ESD/EMP Protection	Air: ± 24 kV, Contact: ± 24 kV
Operating Temperature	-10 to 45° C (14 to 113° F)
Operating Humidity	10 to 90% Noncondensing
Networking Interfaces	
Data/PoE Input Port	(1) 10/100/1000 RJ45 Port
Data Ports	(3) 10/100/1000 RJ45 Ports
Data/PoE Passthrough Port	(1) 10/100/1000 RJ45 Port

Figure G.2: Datasheet of EdgeRouter 6P.

G.1.2 Ubiquiti Networks EdgeRouter 6P


EdgeRouter 6P

Hardware Specifications

ER-6P	
Dimensions	229 x 136.5 x 31.1 mm (9.02 x 5.37 x 1.22 in)
Weight	730 g (1.61 lb)
Max. Power Consumption	16 W (Excludes PoE Output)
Power	External AC Power Adapter, 60W (24V, 2.5A)
Power Input	110 - 240VAC
Button	Reset
LEDs	
Data Ports	Speed/Link/Activity, PoE
SFP Data Port	Link/Activity
Networking Interfaces Management	(1) RJ45 Serial Port (6) Ethernet Ports (Default eth0)
Networking	(5) 10/100/1000 RJ45 Ports with PoE (1) 1 Gbps SFP Port
Processor	4-Core 1 GHz, MIPS64
System Memory	1 GB DDR3 RAM
On-Board Flash Storage	4 GB eMMC, 8 MB SPI NOR
Rack-Mountable	Yes
Operating Temperature	-10 to 50° C (14 to 122° F)
Operating Humidity	10 - 90% Noncondensing
Certifications	CE, FCC, IC

PoE with 24VDC Power Adapter	
PoE Interfaces	(5) 24V Passive PoE Ports, 2-pair (4, 5+; 7, 8-) and 4-pair
Passive PoE Max. Wattage Per Port	24W (24V/1A, 4-pair)
Voltage Range Passive PoE	24V

DATASHEET
EdgeRouter™




7

Figure G.3: Datasheet of EdgeOS.

G.1.3 Ubiquiti Networks Edge OS

EdgeRouter™


DATASHEET



Router Software Specifications

EdgeOS	
Interface/Encapsulation	Ethernet 802.1q VLAN PPPoE GRE IP in IP Bridging Bonding (802.3ad)
Addressing	Static IPv4/IPv6 Addressing DHCP/DHCPv6
Routing	Static Routes OSPF/OSPFv3 RIP/RIPng BGP (with IPv6 Support) IGMP Proxy MPLS
Security	ACL-Based Firewall Zone-Based Firewall Application Identification with Deep Packet Inspection (DPI) NAT
VPN	IPSec Site-to-Site and Remote Access OpenVPN Site-to-Site and Remote Access PPTP Remote Access L2TP Remote Access PPTP Client
Services	DHCP/DHCPv6 Server DHCP/DHCPv6 Relay Dynamic DNS DNS Forwarding VRRP RADIUS Client Web Caching PPPoE Server
QoS	FIFO Stochastic Fairness Queueing Random Early Detection Token Bucket Filter Deficit Round Robin Hierarchical Token Bucket Ingress Policing
Management	Web UI Ubiquiti Network Management System (UNMS) CLI (GUI, Console, SSH, Telnet) SNMP NetFlow LLDP NTP UBNT Discovery Protocol Syslog

Specifications are subject to change. Ubiquiti products are sold with a limited warranty described at: www.ubnt.com/support/warranty
 ©2017-2018 Ubiquiti Networks, Inc. All rights reserved. Ubiquiti, Ubiquiti Networks, the Ubiquiti U logo, the Ubiquiti beam logo, airMAX, EdgeMax, EdgeOS, EdgeRouter, and UNMS are trademarks or registered trademarks of Ubiquiti Networks, Inc. in the United States and in other countries. Apple and the Apple logo are trademarks of Apple Inc., registered in the U.S. and other countries. App Store is a service mark of Apple Inc., registered in the U.S. and other countries. Android, Google, Google Play, the Google Play logo and other marks are trademarks of Google Inc. All other trademarks are the property of their respective owners.


www.ubnt.com

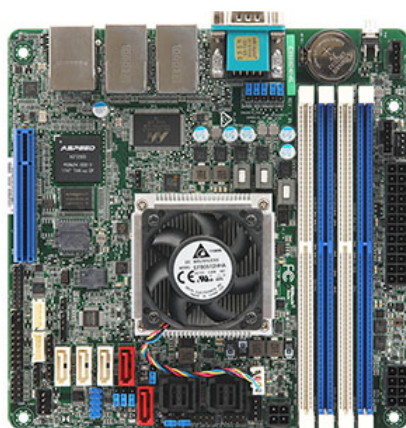
A1022118
8

G.2 End-hosts

The end-hosts in the evaluation testbed, and Home Agent (HA) for Mobile IPv6 (MIPv6) was on a desktop host. More specifically, mini-ITX desktop host with Intel system on a chip (SoC) installed into a mini-ITX case.

The following shows the datasheet for the motherboard.

[About](#) [Contact Us](#) [Products](#)



C3558D4I-4L

- Intel® Atom Processor C3558: 4core, 1
- DDR4 1866/2133/2400 Dual-channel M
RDIMM
- 4 SATA3 6.0Gbps + 1 SATA DOM by Ir
- 1 PCIe3.0 x8



[Specifications](#) [Download](#) [Manual](#) [Memory QVL](#) [HDD QVL](#) [TPM C](#)

Specifications

Physical Status	
Form Factor	- Mini ITX
Dimensions	- 6.7" x 6.7" (170.2 mm x170.2 mm)
Processor System	
CPU	- Intel® Atom C3558 Series Processor: 4core, 16W
Socket	- SOC
System Memory	
Capacity	- 4 x 288-pin DDR4 DIMM slots - Support up to 128GB DDR4 RDIMM (32 GB w/8Gb DRAM) - Support up to 64GB DDR4 UDIMM (16 GB w/8Gb DRAM)
Type	- Dual Channel DDR4 memory technology - Non-ECC, ECC UDIMM: 4GB, 8GB, 16GB - RDIMM: 4GB, 8GB, 16GB, 32GB

DIMM Size Per DIMM	- Dual Channel DDR4 memory technology - Non-ECC, ECC UDIMM: 4GB, 8GB, 16GB - RDIMM: 4GB, 8GB, 16GB, 32GB
DIMM Frequency	- Non-ECC UDIMM: 1600/1866/2133MHz - ECC UDIMM: 1600/1866/2133MHz - RDIMM: 1600/1866/2133MHz
Voltage	- 1.2V
Expansion Slot	
PCIe x 8	- SLOT7: PCIe3.0 x8 (MAX x8 by BIOS setup)
Storage	
SATA Controller	- 5 x SATA3 6.0 Gb/s: 5 x SATA3 ports (SATA_0, SATA_1, SATA_2, SATA_3 and S - Supports up to 8 SATA ports from 2 mini SAS HD connectors (SATA_4_7 and SA each mini SAS HD connector)
Ethernet	
Interface	- Gigabit LAN 10/100/1000 Mb/s
LAN Controller	- Marvell 88E1543(4L) - Supports Wake-On-LAN - Supports Energy Efficient Ethernet 802.3az - Supports Dual LAN with Teaming function - Supports PXE - LAN1 Supports NCSI
Management	
BMC Controller	- ASPEED AST2500 : IPMI (Intelligent Platform Management Interface) 2.0 with Iik
IPMI Dedicated GLAN	- 1 x Realtek RTL8211E for dedicated management LAN
Features	- Watch Dog - NMI
Graphics	
Controller	- ASPEED AST2500
VRAM	- DDR4 64MB
Output	- Supports D-Sub with max. resolution up to 1920x1200 @ 60Hz
Rear Panel I/O	
VGA Port	- 1 x D-sub
USB 3.2 Gen1 Port	- 2
Lan Port	- 4 x RJ45 Gigabit Ethernet LAN ports - 1 x RJ45 Dedicated IPMI LAN port - LAN Ports with LED (ACT/LINK LED and SPEED LED)
Serial Port	- 1
UID Button/UID LED	- 1
Internal Connector	

Auxiliary Panel Header	- 1 (includes chassis intrusion, location button & LED, front LAN LED, and event lo
TPM Header	- 1
IPMB Header	- 1
Fan Header	- 4 pins x4 (1CPU/2Front/1Rear)
ATX Power	- 1 (24-pin)
DC IN	- 1 (8-pin) 12v
USB 2.0 Header	- 1 (support 1 USB port or 2 USB ports on header. The port number controlled by l
Front Panel	- 1
SATA Power Connector	- 1 (4-pin)
System BIOS	
BIOS Type	- 128Mb AMI UEFI Legal BIOS
BIOS Features	- Plug and Play (PnP) - ACPI 1.1 Compliance Wake Up Events - SMBIOS 2.3.1 Support - DRAM Voltage Multi-adjustment - ASRock Instant Flash
Hardware Monitor	
Temperature	- CPU/PCH/DDR/LAN/Storage Temperature Sensing - MB/Card side/TR1 Temperature Sensing
Fan	- CPU/Rear/Front Fan Tachometer - CPU Quiet Fan (Allow CPU Fan Speed Auto-Adjust by CPU Temperature) - CPU/Rear/Front Fan Multi-Speed Control
Voltage	- Voltage Monitoring: +12V, +5V, +3.3V, CPU Vcore, DRAM, +BAT, 3VSB, 5VSB
Support OS	
OS	Microsoft® Windows® - Server 2016 (64 bit) - Server 2012 (64 bit) - Server 2012 R2 (64 bit) Linux - RedHat Enterprise Linux Server 6.8 (32 / 64 bit) / 7.2 (64 bit) - CentOS 6.8 (32 / 64 bit) / 7.2 (64 bit) - SUSE Enterprise Linux Server 11 SP4 (32 / 64 bit) / 12 SP1 (64 bit) - Fedora core 24 (64 bit) - Ubuntu 16.04 (64 bit) / 15.10 (64 bit) (AHCI mode) Virtual - VMWare® ESXi 6.0 * Please refer to our website for the latest OS support list.
Environment	

Temperature	Operation temperature: 10°C ~ 35°C / Non operation temperature: -40°C ~ 70°C * Install a CPU Fan when the airflow through the heatsink is below 2CFM.
-------------	--

The specification is subject to change without notice in advance. The brand and product names are trademarks of their respective owners. The product specification is not guaranteed.

© 2021 ASRock Rack Inc. All rights reserved. | Information published on ASRockRack.com is subject to change without notice.

Bibliography

- [1] R. Yanagida and S. N. Bhatti, “Seamless Internet connectivity for ubiquitous communication,” in *PURBA2019, Pervasive Urban Applications Workshop (UBICOMP 2019)*, Sep 2019, p. 12. [Online]. Available: <https://doi.org/10.1145/3341162.3349315>
- [2] S. E. Deering and B. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” Internet Engineering Task Force, Request for Comments RFC 8200, Jul. 2017, num Pages: 42. [Online]. Available: <https://datatracker.ietf.org/doc/rfc8200>
- [3] R. Atkinson and S. N. Bhatti, “Icmp locator update message for the identifier-locator network protocol for ipv6 (ilnrv6),” IRTF, RFC 6743 (E), Nov 2012.
- [4] Z. Wang, A. C. Bovik, and E. P. Simoncelli, “Structural Approaches to Image Quality Assessment,” in *Handbook of Image and Video Processing*. Elsevier, 2005, pp. 961–974. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/B9780121197926501194>
- [5] “IEEE Standard for Local and metropolitan area networks–Part 21: Media Independent Services Framework,” *IEEE Std 802.21-2017 (Revision of IEEE Std 802.21-2008 as amended by IEEE Std 802.21a-2012, IEEE Std 802.21b-2012, IEEE Std 802.21c-2014, and IEEE Std 802.21d-2015)*, pp. 1–314, Apr. 2017, conference Name: IEEE Std 802.21-2017 (Revision of IEEE Std 802.21-2008 as amended by IEEE Std 802.21a-2012, IEEE Std 802.21b-2012, IEEE Std 802.21c-2014, and IEEE Std 802.21d-2015).
- [6] “IEEE Standard for Local and metropolitan area networks–Part 21.1: Media Independent Services,” *IEEE Std 802.21.1-2017*, pp. 1–211, Apr. 2017, conference Name: IEEE Std 802.21.1-2017.

- [7] “IEEE Standard for Local and metropolitan area networks–Part 21: Media Independent Services Framework–Corrigendum 1: Clarification of Parameter Definition in Group Session Key Derivation,” *IEEE Std 802.21-2017/Cor 1-2017 (Corrigendum to IEEE Std 802.21-2017)*, pp. 1–13, Jan. 2018, conference Name: IEEE Std 802.21-2017/Cor 1-2017 (Corrigendum to IEEE Std 802.21-2017).
- [8] “IPNG (ipngwg) -.” [Online]. Available: <https://datatracker.ietf.org/wg/ipngwg/history/>
- [9] S. E. Deering and B. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” Internet Engineering Task Force, Request for Comments RFC 1883, Dec. 1995, num Pages: 37. [Online]. Available: <https://datatracker.ietf.org/doc/rfc1883>
- [10] M. Weiser, “Some Computer Science Issues in Ubiquitous Computing,” *Commun. ACM*, vol. 36, no. 7, pp. 75–84, Jul. 1993. [Online]. Available: <http://doi.acm.org/10.1145/159544.159617>
- [11] B. E. Carpenter, “IP addresses considered harmful,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 65–69, 2014.
- [12] M. O’Dell, “8+8 - An Alternate Addressing Architecture for IPv6,” Internet Engineering Task Force, Internet-Draft draft-odell-8+8-00, Oct. 1996, internet Draft. [Online]. Available: <https://datatracker.ietf.org/doc/html/draft-odell-8+8-00>
- [13] L. Zhang, A. J. Mankin, J. W. S. III, T. Narten, and M. Crawford, “Separating Identifiers and Locators in Addresses: An Analysis of the GSE Proposal for IPv6,” Internet Engineering Task Force, Internet Draft draft-ietf-ipngwg-esd-analysis-05, Oct. 1999, num Pages: 52. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-ipngwg-esd-analysis-05>
- [14] I. Castineyra, N. Chiappa, and M. Steenstrup, “The Nimrod Routing Architecture,” RFC Editor, Tech. Rep. RFC1992, Aug. 1996. [Online]. Available: <https://www.rfc-editor.org/info/rfc1992>
- [15] S. Sevilla and J. Garcia-Luna-Aceves, “A deployable identifier-locator split architecture,” in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, Jun. 2017, pp. 1–9.

- [16] C. E. Perkins, “IP Mobility Support for IPv4,” Internet Engineering Task Force, Request for Comments RFC 3344, Aug. 2002, num Pages: 99. [Online]. Available: <https://datatracker.ietf.org/doc/rfc3344>
- [17] —, “IP Mobility Support for IPv4, Revised,” Internet Engineering Task Force, Request for Comments RFC 5944, Nov. 2010, num Pages: 100. [Online]. Available: <https://datatracker.ietf.org/doc/rfc5944>
- [18] D. B. Johnson, J. Arkko, and C. E. Perkins, “Mobility Support in IPv6,” Internet Engineering Task Force, Request for Comments RFC 6275, Jul. 2011, num Pages: 169. [Online]. Available: <https://datatracker.ietf.org/doc/rfc6275>
- [19] R. Koodli, “Mobile IPv6 Fast Handovers,” Internet Engineering Task Force, Request for Comments RFC 5568, Jul. 2009, num Pages: 51. [Online]. Available: <https://datatracker.ietf.org/doc/rfc5568>
- [20] K. Chowdhury, K. Leung, B. Patil, V. Devarapalli, and S. Gundavelli, “Proxy Mobile IPv6,” Internet Engineering Task Force, Request for Comments RFC 5213, Aug. 2008, num Pages: 92. [Online]. Available: <https://datatracker.ietf.org/doc/rfc5213>
- [21] C. Castelluccia, H. Soliman, L. Bellier, and K. E. Malki, “Hierarchical Mobile IPv6 Mobility Management (HMIPv6),” Internet Engineering Task Force, Request for Comments RFC 4140, Aug. 2005, num Pages: 29. [Online]. Available: <https://datatracker.ietf.org/doc/rfc4140>
- [22] P. Thubert, A. Petrescu, R. Wakikawa, and V. Devarapalli, “Network Mobility (NEMO) Basic Support Protocol,” Internet Engineering Task Force, Request for Comments RFC 3963, Jan. 2005, num Pages: 33. [Online]. Available: <https://datatracker.ietf.org/doc/rfc3963>
- [23] I. S. Alsukayti and C. Edwards, “Multihomed Mobile Network Architecture,” in *2015 IFIP Networking Conference (IFIP Networking)*, May 2015, pp. 1–9.
- [24] “Distributed Mobility Management (dmm) -.” [Online]. Available: <https://datatracker.ietf.org/wg/dmm/about/>

- [25] J. Iyengar and M. Thomson, “QUIC: A UDP-Based Multiplexed and Secure Transport,” Internet Engineering Task Force, Request for Comments RFC 9000, May 2021, num Pages: 151. [Online]. Available: <https://datatracker.ietf.org/doc/rfc9000>
- [26] E. Nordmark and M. Bagnulo, “Shim6: Level 3 Multihoming Shim Protocol for IPv6,” IETF, RFC RFC5533, Jun. 2009. [Online]. Available: <https://www.rfc-editor.org/info/rfc5533>
- [27] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, “The Locator/ID Separation Protocol (LISP),” Internet Engineering Task Force, Request for Comments RFC 6830, Jan. 2013, num Pages: 75. [Online]. Available: <https://datatracker.ietf.org/doc/rfc6830>
- [28] O. Troan, D. Miles, S. Matsushima, T. Okimoto, and D. Wing, “IPv6 Multihoming without Network Address Translation,” RFC Editor, Tech. Rep. RFC7157, Mar. 2014. [Online]. Available: <https://www.rfc-editor.org/info/rfc7157>
- [29] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, “RFC6824—TCP Extensions for Multipath Operation with Multiple Addresses,” IETF, RFC RFC6824, Jan. 2013. [Online]. Available: <https://www.rfc-editor.org/info/rfc6824>
- [30] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, “TCP Extensions for Multipath Operation with Multiple Addresses,” RFC Editor, Tech. Rep. RFC8684, Mar. 2020. [Online]. Available: <https://www.rfc-editor.org/info/rfc8684>
- [31] R. Stewart, “Stream Control Transmission Protocol,” RFC Editor, Tech. Rep. RFC4960, Sep. 2007. [Online]. Available: <https://www.rfc-editor.org/info/rfc4960>
- [32] R. Stewart, M. Tuexen, and G. Camarillo, “Security Attacks Found Against the Stream Control Transmission Protocol (SCTP) and Current Countermeasures,” RFC Editor, Tech. Rep. RFC5062, Sep. 2007. [Online]. Available: <https://www.rfc-editor.org/info/rfc5062>
- [33] D. Farinacci, D. Lewis, D. Meyer, and C. White, “LISP Mobile Node,” Internet Engineering Task Force, Internet Draft draft-meyer-lisp-mn-16, Dec. 2016, num Pages: 22. [Online]. Available: <https://datatracker.ietf.org/doc/draft-meyer-lisp-mn-16>

- [34] —, “LISP Mobile Node,” Internet Engineering Task Force, Internet Draft draft-ietf-lisp-mn-10, Aug. 2021, num Pages: 25. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-lisp-mn>
- [35] M. Isah, S. Simpson, and C. Edwards, “An Improved LISP Mobile Node Architecture,” *Journal of Network and Computer Applications*, vol. 118, pp. 29–43, Sep. 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S108480451830105X>
- [36] A. Dhraief and N. Montavont, “Toward Mobility and Multihoming Unification- The SHIM6 Protocol: A Case Study,” in *2008 IEEE Wireless Communications and Networking Conference*, Mar. 2008, pp. 2840–2845, iSSN: 1558-2612.
- [37] M. S. Rahman and M. Atiquzzaman, “SEMO6 - a multihoming-based seamless mobility management framework,” in *MILCOM 2008 - 2008 IEEE Military Communications Conference*, Nov. 2008, pp. 1–7, iSSN: 2155-7586.
- [38] R. Moskowitz and P. Nikander, “Host Identity Protocol (HIP) Architecture,” IETF, RFC RFC4423, May 2006. [Online]. Available: <https://www.rfc-editor.org/info/rfc4423>
- [39] R. Moskowitz, T. Heer, P. Jokela, and T. Henderson, “Host Identity Protocol Version 2 (HIPv2),” RFC Editor, Tech. Rep. RFC7401, Apr. 2015. [Online]. Available: <https://www.rfc-editor.org/info/rfc7401>
- [40] T. Henderson, C. Vogt, and J. Arkko, “Host Mobility with the Host Identity Protocol,” RFC Editor, Tech. Rep. RFC8046, Feb. 2017. [Online]. Available: <https://www.rfc-editor.org/info/rfc8046>
- [41] —, “Host Multihoming with the Host Identity Protocol,” RFC Editor, Tech. Rep. RFC8047, Feb. 2017. [Online]. Available: <https://www.rfc-editor.org/info/rfc8047>
- [42] R. Moskowitz and M. Komu, “Host Identity Protocol Architecture,” Internet Engineering Task Force, Request for Comments RFC 9063, Jul. 2021, num Pages: 41. [Online]. Available: <https://datatracker.ietf.org/doc/rfc9063>

- [43] P. Nikander, M. Komu, and T. Henderson, “Using the Host Identity Protocol with Legacy Applications,” Internet Engineering Task Force, Request for Comments RFC 5338, Sep. 2008, num Pages: 14. [Online]. Available: <https://datatracker.ietf.org/doc/rfc5338>
- [44] J. Laganier and L. Eggert, “Host Identity Protocol (HIP) Rendezvous Extension,” RFC Editor, Tech. Rep. RFC5204, Apr. 2008. [Online]. Available: <https://www.rfc-editor.org/info/rfc5204>
- [45] C. Raiciu, D. Niculescu, M. Bagnulo, and M. J. Handley, “Opportunistic mobility with multipath TCP,” in *Proceedings of the sixth international workshop on MobiArch*, ser. MobiArch ’11. New York, NY, USA: Association for Computing Machinery, Jun. 2011, pp. 7–12. [Online]. Available: <https://doi.org/10.1145/1999916.1999919>
- [46] Y. Sun, Y. Cui, W. Wang, T. Ma, Y. Ismailov, and X. Zheng, “Mobility support in Multipath TCP,” in *2011 IEEE 3rd International Conference on Communication Software and Networks*, May 2011, pp. 195–199.
- [47] Y. Liu, Y. Ma, C. Huitema, Q. An, and Z. Li, “Multipath Extension for QUIC,” Internet Engineering Task Force, Internet Draft draft-liu-multipath-quic-03, Mar. 2021, num Pages: 20. [Online]. Available: <https://datatracker.ietf.org/doc/draft-liu-multipath-quic>
- [48] K. Popat and V. V. Kapadia, “Multipath TCP Security Issues, Challenges and Solutions,” in *Information, Communication and Computing Technology*, ser. Communications in Computer and Information Science, M. Bhattacharya, L. Kharb, and D. Chahal, Eds. Cham: Springer International Publishing, 2021, pp. 18–32.
- [49] C. Paasch and O. Bonaventure, “Multipath TCP: Decoupled from IP, TCP is at last able to support multihomed hosts.” *Queue*, vol. 12, no. 2, pp. 40–51, Feb. 2014. [Online]. Available: <https://dl.acm.org/doi/10.1145/2578508.2591369>
- [50] Cisco, “Cisco Annual Internet Report (2018–2023),” Cisco Systems Inc. (USA), Tech. Rep., Mar 2020, updated March 2020. [Online]. Available: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf>

- [51] “Digital Video Broadcasting (DVB); Adaptive media streaming over IP multicast,” ETSI, Technical Specification ETSI TS 103 769, Nov 2020, technical Specification, ETSI TS 103 769. [Online]. Available: <http://dvb.org/?standard=adaptive-media-streaming-over-ip-multicast>
- [52] R. Atkinson and S. N. Bhatti, “Identifier-Locator Network Protocol (ILNP) Architectural Description,” IRTF, RFC 6740 (E), Nov 2012.
- [53] B. Trammell and M. Kuehlewind, “The Wire Image of a Network Protocol,” Internet Architecture Board (IAB), RFC RFC8546, Apr. 2019. [Online]. Available: <https://www.rfc-editor.org/info/rfc8546>
- [54] R. Atkinson and S. N. Bhatti, “Identifier-Locator Network Protocol (ILNP) Engineering Considerations,” IRTF, RFC 6741 (E), Nov 2012.
- [55] R. Atkinson, S. N. Bhatti, and S. Rose, “DNS Resource Records for the Identifier-Locator Network Protocol (ILNP),” IRTF, RFC 6742 (E), Nov 2012.
- [56] R. Atkinson and S. N. Bhatti, “IPv6 Nonce Destination Option for the Identifier-Locator Network Protocol for IPv6 (ILNPv6),” IRTF, RFC 6744 (E), Nov 2012.
- [57] R. Hinden and S. Deering, “Internet Protocol Version 6 (IPv6) Addressing Architecture,” RFC Editor, Tech. Rep. RFC3513, Apr. 2003. [Online]. Available: <https://www.rfc-editor.org/info/rfc3513>
- [58] D. Phoomkiattisak, “Mobility as first class functionality : ILNPv6 in the Linux kernel,” Thesis, University of St Andrews, Jun. 2016. [Online]. Available: <https://research-repository.st-andrews.ac.uk/handle/10023/7915>
- [59] B. Simpson, “Multihoming with ILNP in FreeBSD,” Thesis, University of St Andrews, Jun. 2016. [Online]. Available: <https://research-repository.st-andrews.ac.uk/handle/10023/8681>
- [60] B. Simpson and S. N. Bhatti, “An identifier-locator approach to host multihoming,” in *AINA 2014 - IEEE 28th Intl. Conf. Advanced Information Networking and Applications*, May 2014, pp. 139–147. [Online]. Available: <https://doi.org/10.1109/AINA.2014.22>

- [61] M. Gupta and A. Conta, “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification,” Internet Engineering Task Force, Request for Comments RFC 4443, Mar. 2006, num Pages: 24. [Online]. Available: <https://datatracker.ietf.org/doc/rfc4443>
- [62] T. Narten, T. Jinmei, and S. Thomson, “IPv6 Stateless Address Autoconfiguration,” Internet Engineering Task Force, Request for Comments RFC 4862, Sep. 2007, num Pages: 30. [Online]. Available: <https://datatracker.ietf.org/doc/rfc4862>
- [63] T. Narten and S. Thomson, “IPv6 Stateless Address Autoconfiguration,” Internet Engineering Task Force, Request for Comments RFC 2462, Dec. 1998, num Pages: 25. [Online]. Available: <https://datatracker.ietf.org/doc/rfc2462>
- [64] W. A. Simpson, T. Narten, E. Nordmark, and H. Soliman, “Neighbor Discovery for IP version 6 (IPv6),” Internet Engineering Task Force, Request for Comments RFC 4861, Sep. 2007, num Pages: 97. [Online]. Available: <https://datatracker.ietf.org/doc/rfc4861>
- [65] D. Phoomikiattisak and S. N. Bhatti, “End-To-End Mobility for the Internet Using ILNP,” *Wireless Communications and Mobile Computing*, vol. 2019, no. Article ID 7464179, p. 29, Apr 2019. [Online]. Available: <https://doi.org/10.1155/2019/7464179>
- [66] S. N. Bhatti, D. Phoomikiattisak, and B. Simpson, “IP without IP addresses,” in *AINTEC 2016 - 12th Asian Internet Engineering Conf.*, Nov/Dec 2016, pp. 41–48. [Online]. Available: <https://doi.org/10.1145/3012695.3012701>
- [67] D. Phoomikiattisak and S. N. Bhatti, “Control Plane Handoff Analysis for IP Mobility,” in *WMNC 2016 - 9th IFIP Wireless and Mobile Networking Conf.*, Jul 2016, pp. 65–72. [Online]. Available: <https://doi.org/10.1109/WMNC.2016.7543931>
- [68] S. N. Bhatti, R. Yanagida, K. Shezhad, and D. Phoomikiattisak, “ilnp-public-1,” Sep. 2019, first software release of ILNPv6 prototype implementation containing mobility mechanism. [Online]. Available: <https://ilnp.github.io/ilnp-public-1/>

- [69] N. Moore, “Optimistic Duplicate Address Detection (DAD) for IPv6,” Internet Engineering Task Force, Request for Comments RFC 4429, Apr. 2006, num Pages: 17. [Online]. Available: <https://datatracker.ietf.org/doc/rfc4429>
- [70] M. Shreedhar and G. Varghese, “Efficient fair queueing using deficit round robin,” *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 4, pp. 231–242, Oct. 1995. [Online]. Available: <https://doi.org/10.1145/217391.217453>
- [71] R. Atkinson and S. N. Bhatti, “ICMP Locator Update Message for the Identifier-Locator Network Protocol for IPv4 (ILNPv4),” IRTF, RFC 6745 (E), Nov 2012.
- [72] O. Ejembi and S. N. Bhatti, “Help Save The Planet: Please Do Adjust Your Picture,” in *MM 2014 - 22nd ACM Intl. Conf. Multimedia*, Nov 2014, pp. 427–436, open Access. [Online]. Available: <https://doi.org/10.1145/2647868.2654897>
- [73] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, “Image Quality Assessment: From Error Visibility to Structural Similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr 2004. [Online]. Available: <http://ieeexplore.ieee.org/document/1284395/>
- [74] “VQMT: Video Quality Measurement Tool.” [Online]. Available: <https://www.epfl.ch/labs/mmspg/downloads/vqmt/>
- [75] R. Hahling, “VQMT - Video Quality Measurement Tool,” Aug. 2021, original-date: 2013-09-09T08:01:57Z. [Online]. Available: <https://github.com/rolinh/VQMT>
- [76] F. Gont, A. Cooper, D. Thaler, and W. S. LIU, “Recommendation on Stable IPv6 Interface Identifiers,” Internet Engineering Task Force, Request for Comments RFC 8064, Feb. 2017, num Pages: 9. [Online]. Available: <https://datatracker.ietf.org/doc/rfc8064>
- [77] S. N. Bhatti, G. Haywood, and R. Yanagida, “End-To-End Privacy for Identity & Location with IP,” in *2nd Workshop on New Internetworking Protocols, Architecture and Algorithms 2021*. Virtual event: IEEE, Nov 2021.