Cooperative Resource Pooling in Multihomed Mobile Networks

Richard Withnell

School of Computing and Communications

This dissertation is submitted for the degree of Doctor of Philosophy

Abstract

The ubiquity of multihoming amongst mobile devices presents a unique opportunity for users to co-operate, sharing their available Internet connectivity, forming multihomed mobile networks on demand. This model provides users with vast potential to increase the quality of service they receive. Despite this, such mobile networks are typically underutilized and overly restrictive, as additional Internet connectivity options are predominantly ignored and selected gateways are both immutable and incapable of meeting the demand of the mobile network. This presents a number of research challenges, as users look to maximize their quality of experience, while balancing both the financial cost and power consumption associated with utilizing a diverse set of heterogeneous Internet connectivity options. In this thesis we present a novel architecture for mobile networks, the contribution of which is threefold. Firstly, we ensure the available Internet connectivity is appropriately advertised, building a routing overlay which allows mobile devices to access any available network resource. Secondly, we leverage the benefits of multipath communications, providing the mobile device with increased throughput, additional resilience and seamless mobility. Finally, we provide a multihomed framework, enabling policy driven network resource management and path selection on a per application basis. Policy driven resource management provides a rich and descriptive approach, allowing the context of the network and the device to be taken into account when making routing decisions at the edge of the Internet. The aim of this framework, is to provide an efficient and flexible approach to the allocation of applications to the optimal network resource, no matter where it resides in a mobile network. Furthermore, we investigate the benefits of path selection, facilitating the policy framework to choose the optimal network resource for specific applications. Through our evaluation, we prove that our approach to advertising Internet connectivity in a mobile network is both efficient and capable of increasing the utilization of the available network capacity. We then

demonstrate that our policy driven approach to resource management and path selection can further improve the users quality of experience, by tailoring network resource usage to meet their specific needs.

Declaration

This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except where specified in the text. This dissertation is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any other university. This dissertation does not exceed the prescribed limit of 60,000 words.

Richard Withnell
December, 2015

Acknowledgements

Foremost, I would like to express my deepest thanks to my supervisor, Dr. Christopher Edwards, for the consistent guidance and tutelage over the last four years. Without his time and patience none of this would have been possible. Besides my supervisor, I would especially like to thank the members of my thesis committee, Dr. Utz Roedig and Dr. Michael Welzl, for taking their time to review my thesis, and for providing valuable insight and feedback during my defence.

I am incredibly grateful to Dr. Ben McCarthy, who provided me with much needed direction in the early stages of the PhD and was an excellent source of guidance and knowledge throughout. I would also like to thank those in the Networking Group, that I have worked with closely over the years: Ibrahim Alsukayti, Musab Isah, Yusuf Sani, and Hayat Kara. With whom I have had countless discussions, helping to shape my work. I also need to express my sincere gratitude to Matthew Broadbent and Oliver Bates, for both the technical discussions, and for reminding me that sometimes the best thing for your work is to take a break from it.

Finally, it's imperative that I thank my family for their unconditional support. Especially my parents, Steve and Janet Withnell, who have always been there, providing endless love, support, and encouragement. Without such a positive influence, I would never have accomplished this much.

Contents

List of Figures

List of Tables

Glossary

AAA Anonymity Authentication and Accounting. 45

ABC Always Best Connected. 50

AODV Ad-hoc On-Demand Distance Vector. 38

API application programming interface. 128

BALIA Balanced Link Adaptation. 32

BGP Border Gateway Protocol. 101

CDPD Cellular Digital Packet Data. 20

CMT Concurrent Multipath Transfer. 10

CoNes Collaborative Network Sharing. 45

cTCP Concurrent-TCP. 23

DBAS Deployable Bandwidth Aggregation System. 26

DCE Direct Code Execution. 111

ECMP Equal Cost Multipath Routing. 101

FTP File Transfer Protocol. 26

ICMP Internet Control Message Protocol. 91

INDAPSON Incentive Data Plan Sharing System Based on Self-Organizing Network. 44

IP Internet Protocol. 17

ISDN Integrated Services Digital Network. 20

ISP Internet Service Provider. 47

LACP Link Aggregation Control Protocol. 19

LCP Link Control Protocol. 20

LIA Linked Increases Algorithm. 31

MAP Multipath Advertisement Protocol. 14

MAPD Multipath Advertisement Protocol Daemon. 110

MPRTP Multipath-RTP. 25

MPTCP Multipath-TCP. 10

MTU Maximum Transmission Unit. 20

NAT Network Address Translation. 21

NS3 Network Simulator 3. 147

OLIA Opportunistic Linked Increases Algorithm. 31

OLSR Optimized Link State Routing. 36

PAN Personal Area Network. 56

PCA Principal Component Analysis. 15

PPP Point to Point Protocol. 20

pSockets Parallel Sockets. 26

pTCP Parallel-TCP. 23

QoE Quality of Experience. 10

QoS Quality of Service. 12

RPC Remote Procedure Call. 92

RTCP Real-time Transport Control Protocol. 25

RTP Real-time Transport Protocol. 25

RTT Round Trip Time. 25

SAW Simple Additive Weighting. 102

SCTP Stream Control Transmission Protocol. 22

SDN Software Defined Networking. 45

SHIM6 Site Multihoming by IPv6 Intermediation. 21

SSH Secure Shell. 26

TCP Transport Control Protocol. 17

TRUMP Trusted Mobile Platform. 12

UDP User Datagram Protocol. 19

VoIP Voice over IP. 10

VPN Virtual Private Network. 45

WiSE Wireless SCTP Extension. 54

wVegas Weighted Vegas. 32

CHAPTER 1

Introduction

The Internet is a vast system, linking countless computer networks to one another, providing global connectivity to billions of devices all over the world. Since the turn of the century the demand for bandwidth has exploded; between 2002 and 2014 the total amount of Internet traffic increased by over 16000% and is predicted to increase by an additional 220% by 2019 [29]. While the increase in total Internet traffic is immense, it is now skewed towards mobile devices with consumption in this domain predicted to grow by over 800% by 2019 [30]. The expansion in the mobile domain has been predicated by an influx in smart phones, tablets, and ultra-books, allowing users to consume high bandwidth services on the move. With large screens, fast processors, and limited storage capabilities, users are streaming video and audio, browsing the web, and backing up data remotely, everywhere and anywhere. This in part is facilitated by the increased availability of Internet access; as current smart devices are typically equipped with a range of different access technologies to ensure connectivity is always at the users fingertips. With the amount of diversity in accessing the Internet, always-on connectivity is becoming the norm instead of the exception. In [11], an inverse relationship between WiFi and Cellular availability was discovered across three different cities in the USA, leading to a total network coverage of 91%. Given the availability and heterogeneity of Internet connectivity, this thesis considers the problem of network resource utilisation in a mobile context. The research predominantly focuses on (but is not limited to) how multiple users can cooperate, sharing their available network connectivity to communally improve the quality of service and experience received.

1.1 Motivation

The demand for mobile data has exploded in recent years and is predicted to continue for the foreseeable future [167]. As new access technologies [120] [171] are developed and wireless spectrum is repurposed [119], user applications and services quickly respond by saturating the newly available bandwidth. This is already seen today as cloud services such as Video on Demand adapt, based on bandwidth availability, ranging from Standard to Ultra High Definition streams. The prevalence of such media streaming applications, cloud storage, and real time communication such as Voice over IP (VoIP) [30] will make maintaining a satisfactory Quality of Experience (QoE) across all services increasingly challenging. The solution to this mobile data crunch, cannot be met by traditional advances in access technology or allocation of additional wireless spectrum. Physically, it is not possible to create new spectrum and increasing spectral efficiency is bound by the Shannon Limit [150]. In addition to the mobile data demand, users may experience periods of poor to no connectivity, based on their location in relation to base stations and access points or by reaching pre-allocated bandwidth quotas. Therefore, to alleviate these problems we must do more to improve the utilisation of the bandwidth that is currently available. To counter connectivity and availability issues, devices have become multihomed, which means to possess two or more network interfaces such as WiFi and Cellular, that may be connected simultaneously. However, due to the ubiquity and deficiencies of current multihoming solutions, the network resource available to a mobile device is already under-utilized; access schemes typically function by allocating a single link based on availability as opposed to using quality or cost factors. In the context of a single mobile device, resource pooling [174] protocols such as Multipath-TCP (MPTCP) [60] and Concurrent Multipath Transfer (CMT) [113] can help to improve utilisation by aggregating cellular and WiFi links. Additionally, MPTCP is also able to better manage network resource [122] but the control of which is still too coarse to fully describe how a user may want to allocate their physical access. Moreover, host based protocols are still limited by a given users mobile subscription, and switching cellular providers on the fly is currently not economically viable. To summarise the problem domain, there are three key issues at the edge of the network:

- **Underutilized Capacity** – Multihomed devices often have access to additional connectivity that is not used. This is facilitated by inadequate connectivity models, typically forcing the use of a single Internet connection and optimizing selection based on availability instead of quality.

- **Inadequate Connectivity** – Users face a number of connectivity issues; from poor coverage in specific areas to running out of their data allowance for both WiFi and cellular plans.

- **Increasing Demand** – The shift towards mobile connectivity and consumption requires heavy use of the available wireless access technologies, and is putting unprecedented strain on the cellular network. Addressing the increase in demand traditionally, is becoming too expensive for providers and novel solutions are required.

The increased cost of connectivity has led to providers pushing users to offload to fixed networks, such as WiFi and Femto Cells with wired back-hauls. Alternative schemes have become prominent within the research community, in an effort to enhance mobile data offloading through mobile ad-hoc and opportunistic networks [27]. Novel solutions to the problem of insufficient resource allocation, attempt to crowd source connectivity [13] [181] [166], dynamically establishing mobile networks at the edge of the Internet which allow users to cooperate by sharing their available connectivity. This connectivity model enables users to increase their available bandwidth, improve resilience and can help to alleviate the heavily congested cellular network. These approaches however are still not sufficient in regards to maximizing and optimizing usage of the network resources that are made available to them. Given these observed and well understood problems; the state of the art solutions are still insufficient. To help alleviate these issues, we propose three functional requirements for future mobile devices and networks:

- **Cooperative Resource Pooling** - Hosts in a mobile network should be able to share their Internet connectivity with one another, increasing the total amount of bandwidth available. Sharing can help to alleviate congestion and improve utilisation, when combined with multipath transport protocols such as MPTCP.

- **Policy Driven Resource Management** - Current access models are not descriptive enough to fully meet the users' needs. Access policies are typically crude, prioritizing interfaces to be used independently, regardless of availability or quality. Users should be able to define exactly how and when their different Internet connectivity options are used, based on the context of both the device and the network.

- **Optimal Path Selection** - Many applications have specific network requirements, which can be described in terms of metrics such as bandwidth, delay, jitter and loss. Given a set of network resources, the optimal path should automatically be selected in an effort to improve the QoS and experience that applications receive.

There are three key use cases that we envisage will benefit from the proposed requirements:

Use Case One – Public Infrastructure: It is becoming increasingly common for public spaces, transport and businesses, to be equipped with open Wireless Access Points. The access points typically provide members of the public with access to free or metered WiFi, instead of continued use of their more costly cellular network interfaces. While users typically look for WiFi, in order to offload from their cellular networks, free connectivity options can be congested and of low quality. In [85], a survey of 1375 respondents suggested that 82% of mobile workers found free WiFi access to be "limited, slow, or unreliable". To this end, users routinely share their existing cellular connectivity between their own devices as opposed to using free WiFi or paying for on demand access, ignoring the additional WiFi capacity that is offered. This model can be extended allowing users to connect their devices to benefit from additional diversity and network availability.

Use Case Two – TRUMP: Trusted Mobile Platform (TRUMP)[1] is a collaborative project [47], investigating how mobile technology can be used to improve the management of chronic illness in rural areas across the UK and India. In rural environments, it can be considerably more difficult and time consuming

[1]This PhD has been funded by the TRUMP Project to investigate and provide connectivity in rural areas across the UK and India.

to travel regularly to the nearest health care facility, increasing the burden on vulnerable individuals [10]. The default assumption of the TRUMP project, considers data connectivity to be widely available, allowing users to interact remotely with health care organisations. While this view may be realistic in the UK, with projects such as B4RN [9], providing fiber-to-the-home in rural areas of the north of England, and increasing provision of Cellular networks. In developing countries such as India, the rural population tends to be significantly larger, estimated at around 68% as of 2013, as opposed to 18% of the UK. To meet this demand the TRUMP project envisages the deployment of a rural mesh network, providing Internet access to the community in Cellular, WiFi, and WiMAX black spots, enabling the management of chronic illness remotely. Due to the size of rural communities in India, a single cellular back haul is unlikely to be sufficient to support the expected amount of traffic. To counter-act this, the mesh will need to support a number of heterogeneous access technologies. Furthermore, as the size of a rural village in India can range between less than 50 and more than 10,000 residents [154], the scalability of the mesh network as well as the ability to balance traffic across the available Internet connectivity is of vital importance.

Use Case Three – First Responder Networks: This use case has been drawn from correspondence with a UK fire service. When approaching large scale fires, the area is split into multiple sectors each of which is serviced by a fire engine. Each sector and fire engine has a number of team members associated with it that have application requirements, such as voice and video streaming back to the headquarters. These services are currently unfulfilled due to unsuitable and unreliable network infrastructure. Communication requirements between the sectors and teams are currently serviced by "runners" i.e., team members who's primarily role is to move information between sectors. To address this problem prior work has interconnected each of the fire engines using a wireless mesh, adding a range of cellular links to provide Internet access. This connectivity can then be extended to the team members through the introduction of WiFi access points, which is used to support a diverse set of emergency applications. This use case presents a complex network topology, with applications that require a guaranteed QoS while mobile. Currently, the available connectivity models do not meet these requirements; therefore, it is necessary to improve the utilisation of the available network resources, and optimise the QoS of the active applications.

1.2 Contributions

The initial contribution of this work is the design, implementation and evaluation of the MAP, which goes beyond the traditional use of a single, static Internet connection per network or gateway, and enables multipath protocols and applications to benefit from the dynamic addition of Internet connectivity by actively pushing information regarding the state of multihomed links into the network. The following contributions, improve the usability and efficacy of the proposed MAP with the design and development of a host-centric policy framework, providing a much needed, descriptive approach identifying how and when the available network resources should and can be used. Finally this is combined with a novel path selection algorithm to allocate single path applications to the most appropriate network resource. These contributions can be summarised as a routing overlay, policy framework and path selection algorithm, described in more detail as follows:

1. **Multipath Advertisement Protocol** : We initially design, implement and evaluate an approach for disseminating multihomed information throughout a mobile network. Doing so makes each host in the network aware of the available connectivity options, which can in turn allow the users applications to benefit from increased throughput and resiliency. Additionally, making the host aware of the available Internet connectivity options can enable multipath protocols such as MPTCP to create additional subflows for each connection, no matter where it resides in the mobile network. We believe that dissemination of network resource is necessary, as the mobile device itself is best suited to choosing the most appropriate link or subset of links for a specific application. The reasoning for this is twofold, firstly the host can easily be made aware of the requirements of its applications. Secondly, a gateway is typically unaware of the alternative network resource available to a mobile device, without the introduction of additional and complex signalling protocols.

2. **User Policy Framework** : Mobile devices suffer from inadequate management of multihomed network interfaces, which is further exacerbated when the number of available Internet connectivity options is expanded using the proposed routing overlay. To this end, we propose a framework

with multihoming and multipath communications at the core of the design. This framework allows users to specify exactly the conditions in which their Internet connections can be used, based on the current context of both the mobile device and the network. We believe that by increasing the granularity of control a user has over the use of their network interfaces it is possible to improve both the QoS and experience. We demonstrate how the policy framework can meet the needs of the user, and further more evaluate the QoS and experience benefits that can be drawn from improving the granularity of network resource usage.

3. **Path Selection Algorithm** : A significant portion of traffic may not be appropriately handled by multipath protocols. Real time communications and flows limited by the application (as opposed to the network) may not benefit from splitting their packets over a diverse set of links, and at worst the QoS could be significantly degraded by using multiple paths with vastly different network metrics. To account for the subset of applications that are better suited to single path communications, we present a path selection algorithm using PCA, to statistically choose the best path by removing the correlation that exists between network metrics. This algorithm is built into the policy framework to dynamically allocate applications to the optimal link in the mobile network.

1.3 Thesis Outline

2. Background: In this chapter we focus on the core networking concepts that the presented work is based on, surveying the state of the art in resource pooling. Firstly, we present a layered view of current technologies that leverage multihoming and multiplicity in the network. This is followed by an in-depth introduction to MPTCP, focusing on the core aspects of the protocol in addition to novel path management and congestion control approaches.

3. Crowdsourcing Connectivity: This chapter presents an overview of related work, including multihomed networks, techniques that allow multiple interfaces to be used, and further more how these approaches can allow users to cooperate and take advantage of diversity in network resource. We then reflect

on the direction and success of the related work, and subsequently detail a set of requirements that we believe are necessary to achieve the proposed goals of improving network resource utilisation and increasing QoS and experience.

4. Design: The design chapter presents the specification for the routing overlay, policy framework and path selection algorithm. The focus of this chapter is to address theoretical challenges and provide robust architectural guidelines for implementation of the proposed work on any suitable platform.

5. Implementation: The implementation chapter presents a realisation of the specified design. Detailing a deployable Linux solution of the routing overlay, policy framework and path selection algorithm.

6. Evaluation: We first evaluate the applicability of MPTCP for use across a large number of independent and heterogeneous paths. Subsequently we investigate and evaluate the practicality and scalability of the proposed routing overlay, in a simulation environment. The evaluation then continues, with a real world testbed, initially verifying the simulation results and then extending them, by introducing the policy framework and path selection algorithms, using QoS and experience metrics to determine the level of improvement.

7. Conclusion: In this final chapter, we reflect on the work presented in this thesis. We identify areas of future work that we predict to be of significant interest to the research community, and propose the necessary steps to continue to leverage the power of multihoming and cooperative resource pooling in a mobile context.

CHAPTER 2

Background

In the previous chapter, we outlined a series of problems with the current mobile connectivity model. This model prevents individual users from maximising the utilisation of the available connectivity options, through the combination of inadequate multihoming functionality and sharing capabilities. As multihoming has become the de facto standard for mobile devices there has been extensive research in this area, attempting to aggregate, load balance, and increase resilience across the diverse set of heterogeneous access technologies that are available, progressing towards the resource pooling principle [174]. In this chapter, we present traditional technologies and protocols as well as the state of the art research in the resource pooling domain. Secondly, we present a detailed analysis of Multipath-TCP, as one of the most successful and widely deployed protocols, abiding by the resource pooling principle.

The main point of reference in this chapter is the TCP/IP network stack as shown in Fig. 2.1. This model describes a layered set of abstract functionality, specifying how devices should communicate, which has a fundamental impact on how new resource pooling and connectivity approaches are defined. Customarily new solutions to these problems are either confined to a single layer or spread across multiple layers.

Figure 2.1: Internet Protocol Suite

2.1 Resource Pooling

The deep-rooted focus on resilience, redundancy and load balancing has been imperative to the success and development of the Internet. At its core the Internet is intrinsically a multipath system, providing unprecedented diversity in terms of the number of paths available to service two remote end-points. Despite this, the suite of protocols and technologies that run on top of the network are typically single path, leading to a fundamental gap between the network and the applications. In [174], Wischik et al. present the resource pooling principle, which they describe to be a collection of network resources behaving like a single pooled resource. The main benefits of resource pooling are defined as follows:

1. Increased robustness.
2. Ability to handle traffic surges.
3. Maximize utilisation.

These benefits match closely to the problems that we presented in Chapter 1, due to inadequate support for multihoming in mobile devices and current network stacks. This section presents a layered overview of both traditional and novel techniques that take advantage of multihomed devices, increasing throughput and resilience by utilizing the bandwidth of additional redundant paths. As with many networking problems, this is a challenge that can be solved at any layer in the network stack. Each of the proposed layers and technologies presents its own unique strengths and weaknesses, as there is a fundamental trade-off between the applicability and the ability to optimize each solution. For example, the lower in the stack a solution resides the more technologies and protocols it will encompass; while the higher up in the stack, the solution becomes increasingly tailored to the needs of specific applications. For this section, we focus on the Internet Protocol Suite which is a realisation of the OSI model, considering: the link, network, transport and application layers.

2.1.1 Challenge

As discussed the de facto approach for two hosts to communicate using the TCP/IP stack is single-path. When introducing additional paths that may be

used for the same connection, stream or flow, the differences in link characteristics can have a significant impact on the performance and therefore the proposed solution. The differences in link characteristics that need to be addressed can be categorised as bandwidth and latency heterogeneity as proposed in [52]. In a mobile environment the heterogeneity present between links can be massive; furthermore, the capabilities and characteristics of individual links can also change frequently, as access technologies change based on availability while the potential bit rate depends on the type of modulation and coding which can fluctuate based on link quality. Bandwidth heterogeneity requires that the solution is able to balance packets or traffic over individual links, depending on the available capacity, in order to maximise throughput. TCP achieves this over a single link using congestion control, to increase the rate that packets are sent and dropping it in the event of packet loss. The heterogeneity of latency across paths, can have a significant and detrimental effect on resource pooling, as out of order packets in TCP can lead to the congestion window being cut in half. Latency heterogeneity may also have a negative impact on specific User Datagram Protocol (UDP) applications such as voice communications, if packets are received after the playback point, they become redundant and are dropped, degrading the perceived quality of the application.

2.1.2 Link Layer

The link layer is the lowest in the TCP/IP stack, it is only concerned with protocols that enable communication between directly connected hosts that reside on the same local network. Therefore no routing is required to transmit a packet from one host to another. At this layer bandwidth aggregation typically focuses [81] [37] on a single hop as this is the fixed scope of the available protocols. This is realistically only beneficial if the bottleneck for a hosts connectivity is the immediately attached link or network. Link aggregation as defined in [81] requires support from the directly connected switch, and is managed by the Link Aggregation Control Protocol (LACP) to simplify configuration. 802.11ax specifies that all Ethernet links must share the same speed and duplex settings and additionally transmits all packets for the same flow on the same link, minimising the issue of re-ordering. While Link Aggregation is highly beneficial and widely used to remove bottlenecks that exist between two directly connected hosts, it

is not designed for use in a wider context. Furthermore the re-ordering issues encountered would be magnified, without buffering or appropriate scheduling to account for this.

Many cellular and wireless access technologies rely on Point to Point Protocol (PPP) for their link layer protocol. In an effort to improve the typically poor throughput of these wireless links, there have been a number of extensions proposed to PPP. In [157], one of the first standards is presented for splitting and recombining datagrams across multiple PPP links. This approach requires compatibility at each end-point of the connection and modifies the Link Control Protocol (LCP) to indicate support for multiple links, additionally datagrams are numbered so they can be re-ordered before being passed up to the network layer. The proposed scheduler simply performs a round-robin over the available links. While this work is applicable to multiple PPP links in any context, the focus was originally on exploiting a set of channels within Integrated Services Digital Network (ISDN). Subsequent work addressed these assumptions by specifically targeting PPP in wireless environments. In [158], the heterogeneity issues associated with wide-area wireless access networks are identified. The authors presented their work in the context of a Cellular Digital Packet Data (CDPD) network, however the fundamental problems induced by variable bandwidths and latencys remain the same today. This work uses Multilink PPP as its foundation and extends it with a more appropriate scheduler based on the estimated link quality, representing the estimated throughput. The scheduling algorithm artificially alters the Maximum Transmission Unit (MTU) depending on the perceived throughput for each link in order to balance load. Altering the MTU however does not fully account for the heterogeneity exhibited by multiple wireless links, which can degrade throughput through re-ordering at the transport layer or by extended waiting times at the link layer.

2.1.3 Network Layer

The primary functionality of the network layer is to ensure that two remote end-hosts are able to communicate, through globally addressing hosts and forwarding packets between one or more networks. The most ubiquitous network layer protocol in use today is the IP, which provides connectionless communication as each packet is handled on an individual basis, depending purely on the addressing in-

formation provided. To this end, IP inherently supports multipath and therefore multihomed hosts. As the network layer is the first point in the network stack in which each of the end-points are known (discounting LANs behind Network Address Translation (NAT)) it provides a promising foundation for bandwidth aggregation solutions. Additionally, as all network applications effectively use IP, a network layer solution can provide global bandwidth aggregation for a host, covering applications relying on any transport layer protocol. While this global applicability may be appealing, accounting for the characteristics of different transport layer protocols can be incredibly challenging, in regards to buffering and re-ordering of packets across flows.

One of the first proposals for network layer bandwidth aggregation required the use of IP-in-IP tunnels [132]. For each available network interface, an IP tunnel is established and the transport layer packets are simply striped over them, this provides a straightforward aggregation solution; however, while it is a functional approach, it fails to address the challenges introduced by bandwidth and latency heterogeneity. Furthermore, IP in IP is typically not suited to traversing NAT boxes making it an unsuitable approach for IPv4 and especially mobile devices. The proposal for Site Multihoming by IPv6 Intermediation (SHIM6) [117], presents a more appropriate and feasible IPv6 based solution, requiring a new shim layer be inserted between the IP and transport layer headers. The premise behind the SHIM6 protocol, is to provide IP address agility, allowing applications to change their IP address mid connection and additionally allow different applications to use different paths. The shim layer separates the locator from the identifier, masking the address changes that may occur from the application and preventing connections from breaking. The shim functions by signalling additional addresses that may be used and can subsequently balance connections and flows across the available paths; therefore, spreading the load and taking advantage of path diversity. The SHIM6 protocol however, does not take full advantage of the available paths, for a single connection only one path can be used. Furthermore, path quality is again not accounted for, which may lead to some applications being allocated to sub-optimal paths. Evensen et.al [53] propose a complete network layer bandwidth aggregation solution, using a proxy to aggregate and stripe packets across the available paths. The authors introduce network layer indicators for path quality, such that packets can be transmitted over the most appropriate link. Initially, the latency across all available paths

was equalized, by introducing artificial delay on the lower latency links. While this is sub-optimal, especially in the case of extreme latency heterogeneity, it can improve the re-ordering observed. In subsequent work [55], the authors remove the delay equalizer, in favor of adding sequence numbers to each packet allowing the network layer proxy or host to re-order the packets before being passed on to the transport layer, or transmitted to other hosts.

2.1.4 Transport Layer

The transport layer provides additional functionality on top of the end-to-end communication of the network layer. The introduction of ports in addition to addresses, enables multiple end-points to be created for each end-host allowing a number of applications to run simultaneously. Therefore the transport layer is responsible for ensuring that data is delivered appropriately to each application, leading to a range of transport layer protocols to suit different requirements. Typically TCP provides a reliable byte stream, while UDP offers a connectionless best effort service. Bandwidth aggregation approaches at the transport layer can be split into two main categories. Those which extend current transport protocols and those which present new transport protocols, in this thesis we will focus on extended transport protocol. In regards to this catagory, the main focus to date has been on TCP and Stream Control Transmission Protocol (SCTP). Due to the use of congestion control by TCP and SCTP, the transport layer is provided with additional information regarding the associated network metrics for each path that is currently in use. In terms of TCP this can be represented by the congestion window and the round trip time, providing an estimate of the available bandwidth and latency. These metrics are imperative to the operation of TCP (likewise with congestion control enabled SCTP) and can help to provide a more educated decision as to how packets should be multiplexed over the set of available paths.

2.1.4.1 TCP

Transport Control Protocol (TCP) is the most widely used transport protocol in existence, and accounts for the vast majority of Internet traffic today. The scalability of TCP has ensured its success as the de facto transport layer protocol since its inception. While it has evolved over the years, the fundamental nature

of the protocol remains the same. It provides a connection oriented, reliable, ordered and error-checked byte stream. Through the use of flow and congestion control, a TCP connection is able to prevent both the end-hosts and the network from becoming overloaded. Due to the pervasive nature of TCP, it has seen a number of extensions proposed to bridge the gap between the traditional single path transport layer and the multipath network.

One of the first multipath extensions proposed was Parallel-TCP (pTCP) [74], this work introduced the idea of associating multiple TCP flows with a single connection, as each flow is provided with its own congestion control algorithm, the work presents a decoupled approach to congestion control allowing the maximum bandwidth of each path to be used. Subsequently mTCP [183] used a similar technique, using multiple TCP subflows to stripe packets across a set of available paths. The authors however, presented a shared congestion control algorithm, attempting to detect shared bottlenecks to maintain fairness. Bottleneck detection is determined by the correlation of fast-retransmits across paths. Concurrent-TCP (cTCP)[41], proposed in 2006, presents another alternative approach, by using a single global congestion window for all interfaces. During the three-way handshake cTCP provides a list of the available interfaces, which allows packets to be transmitted and used over multiple paths instead of establishing multiple independent connections. The more recent MPTCP proposal [60], has seen significant traction in comparison to other efforts and has been implemented and deployed in a number of high profile devices and applications, such as Apple's iOS [6]. The design decisions present in the MPTCP proposal learn from previous efforts and keep deployability at the heart of the work. Instead of extending TCP flows, MPTCP relies purely on the already present option space within the header for signalling. Furthermore, MPTCP proposes a semi-coupled congestion control algorithm differing from pTCP's and mTCP's, uncoupled and coupled approaches respectively. The use of a semi-coupled congestion control algorithm, provides MPTCP with a trade-off between maximising throughput and quickly responding to congestion on each path.

2.1.4.2 SCTP

The Stream Control Transmission Protocol (SCTP) [161] is a flexible alternative to more traditional transport protocols, as it attempts to fill the middle ground

between the rigid connection oriented byte stream of TCP and the connectionless best effort approach of UDP. As SCTP was developed much more recently than both TCP and UDP, multihoming was considered as part of the fundamental protocol design, allowing each connection to support a number of IP addresses. Due to the multihoming principals incorporated into SCTP, it makes a promising foundation for bandwidth aggregation solutions. The current standard for SCTP only uses multihoming to provide resilience and load balancing, however complete bandwidth aggregation schemes are not considered. The simultaneous use of multiple paths has however been an area of interest since its inception. As SCTP provides an underlying multihomed architecture, the focus of bandwidth aggregation and resource pooling has been on appropriately adapting transmission of packets and acknowledgements over the most appropriate link, to address bandwidth and latency heterogeneity.

CMT [113] proposed an extension to SCTP introducing multipath capabilities. As with pTCP, each path is allocated its own congestion window, which is decoupled, such that CMT will be more aggressive in the presence of a shared bottleneck. To account for variability in round trip times and minimise the impact of re-ordering, the authors propose the removal of erroneous fast re-transmits. Furthermore acknowledgements are delayed according to the SCTP standard [161], they are also delayed in the event of re-ordering. Moreover, the acknowledgements are extended with extra path based information, allowing the recipient to determine the difference between loss and re-ordering. The evaluation of CMT, presents promising performance improvements at the transport layer. However, the evaluation presented in [113] does not account for paths with diverse characteristics, only modifying the percentage of loss experienced on each path. More recent work has presented modifications to CMT which account for heterogeneous paths [43] [42] [2]. In [2] the authors propose that the problems that arise when using heterogeneous paths can be presented as a set of blocking issues, as buffers at both the sender and receiver are limited while waiting on slower paths. To address the buffer blocking problem, the authors propose a buffer splitting approach that balances the amount of outstanding data that can be sent on each individual path. In [103], cmpSCTP is proposed; as with CMT, cmpSCTP uses a decoupled congestion window for each path. cmpSCTP differs from CMT, by introducing a two level sequence number space, providing sequence numbers for both the individual paths and the connection as a whole, this simplifies the identification of

re-ordering in comparison to the previously discussed CMT approach, removing the need for previously proposed extensions.

2.1.4.3 RTP

The Real-time Transport Protocol (RTP) [146] is typically used to support audio and video applications, enabling real time communications. RTP provides an abstraction over the underlying transport layer, such as TCP or UDP, dedicated to the principle of application level framing; which allows RTP to carry a range of media types, and for new types to be implemented without modifying the core protocol. Real-time Transport Control Protocol (RTCP) provides out-of-band control information and statistics about the communications such as loss percentages and delay. Real time applications face a different set of challenges to those typically addressed by TCP and SCTP, especially in the multipath and bandwidth aggregation domains. Due to the nature of real time communications and heterogeneous network paths, it becomes incredibly challenging to ensure multiple paths are used appropriately. In the case of TCP and SCTP simple solutions may wait for a packet to be received over a slower link, however for real time communications this may not be a viable option as the information can quickly become redundant.

Despite the challenges associated with exploiting multiple paths for RTP, there have been research efforts aiming to address the previously discussed problems. In [155], the authors propose multipath extensions to RTP, utilizing both the core protocol in addition to RTCP. This work presents the base requirements and design decisions that are necessary to implement a deployable Multipath-RTP (MPRTP). MPRTP focuses on a similar set of goals to MPTCP, in that the authors attempt to balance congestion across links while maintaining a constant bit rate stream, without degrading the users QoS. MPRTP creates a set of subflows for each desired path; and subsequently creates an RTCP stream for each subflow, to monitor the quality in terms of Round Trip Time (RTT), jitter, and lost or discarded packets. The quality information associated with each path is then used by the MPRTP scheduler. The receiver additionally implements a "de-jitter" buffer to reorder packets, attempting to improve the quality of playback. When there are a large number of paths available, the authors propose grouping paths with similar network metrics together, which are then ranked such that

paths with high bandwidths and low delays are preferred. The base concept for the MPRTP scheduling approach is to send the next packet over the link with the highest $\frac{bandwidth}{delay}$, highest bandwidth, and the lowest loss rate. If any packets are lost, they are re-transmitted over the path that is currently assumed to be the best, according to the previous criteria. If a path becomes congested, the scheduler will attempt to shift traffic away in an effort to reduce congestion on these links.

2.1.5 Application Layer

The application layer typically provides a specific service, which is dependent on the underlying layers to facilitate communication, such as the File Transfer Protocol (FTP) or Secure Shell (SSH). The type of application at this layer dictates precisely the network requirements to receive the optimal QoS, which is not possible at the transport layer. For example, FTP benefits from being allocated as much bandwidth as possible, while the interactive nature of SSH sessions can be improved through faster response times. Application layer solutions commonly take these factors into account, optimizing bandwidth aggregation techniques for a specific domain [54], losing generality in favor of improved performance. One of the significant issues with this approach is that applications will typically need to be modified or created to support a specific bandwidth aggregation approach, neglecting legacy applications that may not be upgradable.

The authors of PSockets [156] define the challenges associated with tuning TCP to match the underlying network. Typically manual tuning is necessary choosing the correct congestion control algorithm and buffer sizes for optimal performance. The authors negate this manual tuning process, by creating multiple Parallel Sockets (pSockets), each of which is associated with an underlying TCP connection. To this end, the authors prove that striping data across multiple sockets at the application layer can improve performance, inline with optimal TCP tuning. While the authors only consider the problem of improving performance over a single path, the PSockets concept could easily be extended to a multihomed device, being well suited to quickly downloading large files, due to the increase in throughput. The Deployable Bandwidth Aggregation System (DBAS) [70] is a bandwidth aggregation middleware, that attempts to estimate both the requirements of the application and the available network interfaces to

maximise the efficacy of the solution. For single connection oriented transport protocols, the DBAS attempts to allocate the optimal link for the application, such that all packets are transmitted over the same interface, which seamlessly supports legacy servers. If the server is DBAS capable, then a packet-oriented approach can be used, in which individual packets for the same application can be scheduled over different network interfaces or paths. The authors propose a linear algorithm, that aims to choose the optimal interface, however the parameters of the presented optimisation function, neglects real time characteristics such as latency and jitter, instead focusing on bandwidth and system load. Additionally, the estimation of application requirements also focuses on bandwidth over more find grained characteristics, which may be necessary to fully describe the expected behaviour.

In [54], Evensen et.al present an application specific bandwidth aggregation scheme. The authors propose a multihomed extension to DAVVI [86] (HTTP Adaptive Streaming), enabling the use of multiple network interfaces. The multihomed extension splits video segments into smaller subsegments that are retrieved across different paths simultaneously. The size of subsegments is calculated on-the-fly according to the available throughput for each network path, improving the adaptivity of the DAVVI client. This improvement, is represented by increasing the users QoE, downloading higher quality segments and dropping the total number of interruptions during playback. In [170] a multipath rate controller was developed and evaluated; this solution relied on TCP at the transport layer to split data over multiple independent paths aggregating at a single homed host as the receiving end point. Alternatively it was proposed in [184] that transport layer multipath protocols such as MPTCP are overly complex for certain applications such as file transfer; by implementing more specific protocols such as the suggested Self-adapted Rescheduling Reliable Multipath Transfer Protocol, light weight and deployable bandwidth aggregation can be achieved.

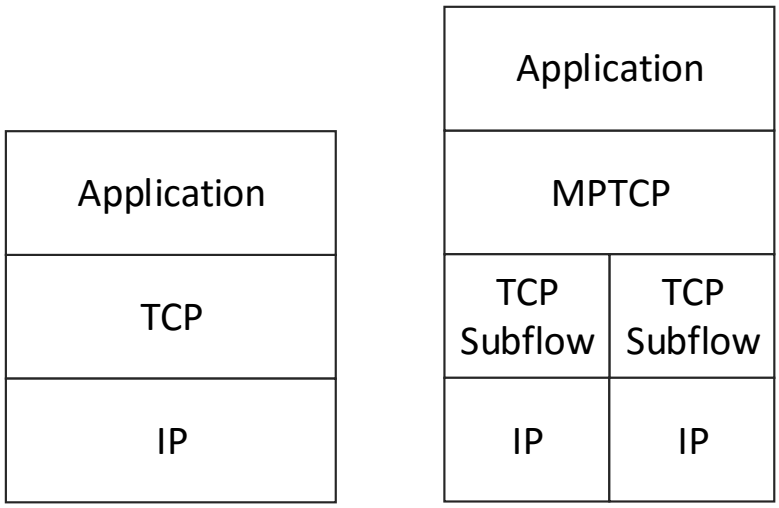
2.1.6 Summary

From this layered survey of bandwidth aggregation schemes, it is obvious that there will not be a single standard enabling an end-host to use all of the available network resource. However, we believe that the transport layer is realistically the lowest point in the network stack that effective bandwidth aggregation schemes

should be explored. While the network layer is usable, we believe that bandwidth aggregation is not a function that should be supported at this point in the stack, due to the wide range of arbitrary applications and protocols that are dependent on it. Additionally, the network layer solutions presented typically rely on the use of a proxy, which could lead to them following in the footsteps of Mobile IP, with a lack of uptake due to additional infrastructure and cost. At the transport layer, the set of path metrics and measurements that are required for effective bandwidth aggregation begins to be exposed inherently, making it much more suitable. Furthermore the types of applications using specific transport layer protocols become better defined. The common factor that essentially all of the discussed technologies share, is the reliance on multiple addresses to inform the host that they are multihomed and addressable across the network. Taking this common factor into account at the design stage will theoretically allow for any of the presented and future technologies to be used for cooperative resource pooling. Of all the presented bandwidth aggregation approaches, the majority focus on aggregation at the end-host, therefore additional paths that exists one or more hops away are neglected and could limit the desired performance and resilience improvements. In [142] and [151] the authors attempt to provide bandwidth aggregation for multiple users through the use of a multihomed gateway. This however, is still limited to aggregating bandwidth based on a single hosts available paths despite passing the benefit on to multiple users.

2.2 Multipath-TCP

In the previous section we presented a layered view of resource pooling and bandwidth aggregation technologies. As we briefly discussed, the most prolific and successful deployment of such a protocol to date is MPTCP. So far the protocol has seen significant uptake with implementations spanning a wide range of systems including, Linux, FreeBSD and Apple's iOS. One of the key factors to the success of MPTCP is transparency, to both the application and the network. Any current applications are able to benefit from multipath communications without modification. Additionally, as MPTCP relies purely on traditional TCP subflows with only additional header options used for signalling, the current network infrastructure is able to handle the extensions seamlessly (the difference between



(a) (b)

Figure 2.2: Relationship between the traditional TCP and MPTCP network stacks.

Option	Value	Description
MP_CAPABLE	0x00	Indicates multipath capability.
MP_JOIN	0x01	A new subflow is joining an established connection.
DSS	0x02	Maps the subflows byte to the matching data sequence.
ADD_ADDR	0x03	Add an address to the connection.
REMOVE_ADDR	0x04	Remove an address from the connection.

Table 2.1: Multipath Specific TCP Options

the TCP and MPTCP network stack is shown in Fig. 2.2). For these reasons we believe that MPTCP, as well as similar transport layer extensions have a future in next generation networks, and will subsequently be a critical component in supporting seamless, heterogeneous mobile connectivity. Therefore for the remainder of this thesis, we will consider MPTCP when making design and implementation decisions, in order to maximise the benefits of bandwidth aggregation and resource pooling. For the remainder of this section, we will provide a more detailed overview of MPTCP. The four main components that MPTCP relies on, is the signalling between hosts, coupled congestion control, path management, and scheduling.

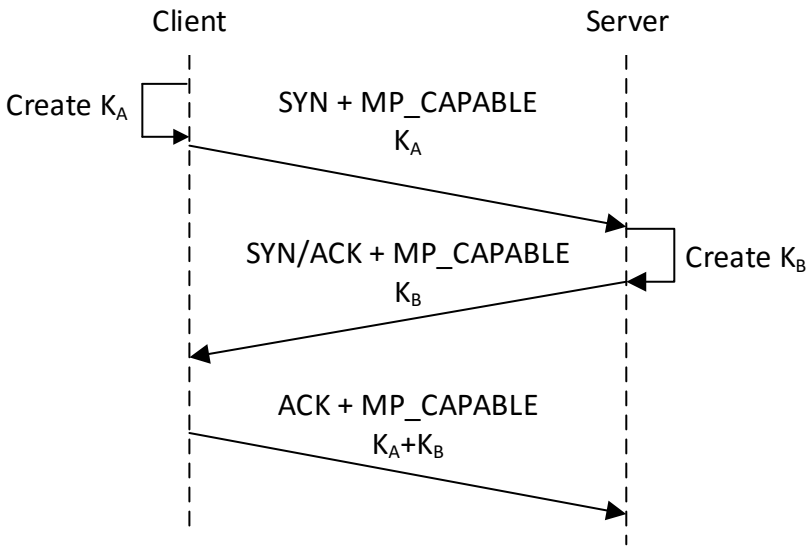


Figure 2.3: Sequence for the initial MPTCP Three Way Handshake

2.2.1 Signalling

To maintain backwards compatibility and provide transparency, MPTCP is founded on creating subflows that look and behave like traditional TCP. The only modification required at the packet level, is the introduction of additional MPTCP options, inserted into the TCP option space. A subset of the core options for controlling an MPTCP connection are shown in Table 2.1 on the previous page. The $MP\_CAPABLE$ option identifies if a host understands MPTCP, similarly to cTCP, if during the three-way handshake, a SYN or $SYN - ACK$ is received without this option, the connection falls back to traditional TCP providing backwards compatibility. During the multipath-capable three-way handshake, a unique key is created which is used to identify the MPTCP connection, shown in Fig. 2.3. The MP_JOIN option is used when a new subflow is added to the MPTCP connection. The new subflow signals the MP_JOIN option during the three-way handshake, along with a token generated from the previously described key, to identify which MPTCP connection the subflow belongs to. Like pTCP, MPTCP creates an additional sequence and ACK number space for the global connection, this includes a Data Sequence Number and Data ACK, which

is passed in the Data Sequence Signal (DSS). The DSS option describes how the data from an individual subflow maps back to the ordering at the connection level and similarly for the connection level acknowledgements. The ADD_ADDR and $REMOVE\_ADDR$ options are used when one or both of the remote end-hosts has additional IP addresses that can be used; for example, the server will indicate an additional address is available through the use of the ADD_ADDR option, the client side will process this option and subsequently attempt to create a new subflow for the announced address.

2.2.2 Congestion Control

The MPTCP working group presents three goals for congestion control algorithms [135] to be used by MPTCP as follows:

1. **Improve throughput** – The MPTCP flow should obtain at least as much throughput as single path TCP.

2. **Do no harm** – At a shared bottleneck, MPTCP should not receive more than its fair share of bandwidth in comparison to a single path TCP flow.

3. **Balance congestion** – MPTCP should move traffic away from the most congested paths.

Inspired by the work of Kelly and Voice in [97], MPTCP proposes the use of a coupled congestion control algorithm to achieve these goals, initially proposing the Linked Increases Algorithm (LIA) [135]. LIA alters the work of Kelly [97], by only coupling the increases, such that for each ACK received on path i the congestion window $cwnd_i$ is increased by Eq. 2.1. In this equation α describes how aggressive the multipath subflow should be in terms of throughput and is tuned appropriately as discussed in [135], negotiating the trade-off between balancing congestion and responsiveness. The work of Khalili et.al [99] describes inefficiencies in the Linked Increase Algorithm, which leads to suboptimal balancing of congestion, effectively breaking Goal 3. The proposed solution is the Opportunistic Linked Increases Algorithm (OLIA), which alters the linked increase phase of LIA. For each ACK on path i, OLIA increases $cwnd_i$ by Eq. 2.2, where I is the set of available paths. The first term of this equation is based purely on [97], while the second term ensures the responsiveness of OLIA and

is calculated according to [99]. Subsequently a number of additional congestion control algorithms have been designed and implemented for MPTCP, including Weighted Vegas (wVegas) and the Balanced Link Adaptation (BALIA). In wVegas [21] the authors propose a delay based congestion control algorithm for MPTCP, extending the work originally proposed in [18] for multiple paths. While the aim of BALIA [169], is to balance the trade-off between the responsiveness and friendliness of MPTCP, providing a middle ground between LIA and OLIA.

$$\frac{\alpha \cdot bytes\_acked \cdot mss_i}{total\_cwnd} \tag{2.1}$$

$$\frac{cwnd_i/rtt_i^2}{(\sum_{p\epsilon I} cwnd_p/rtt_p)^2} + \frac{\alpha_i}{cwnd_i} \tag{2.2}$$

2.2.3 Path Management

The path management functionality of MPTCP determines how, and when subflows should be created over the set of available interfaces. MPTCP currently provides a simple interface controlling the multipath capability of each network interface. Three different options are available, a specific interface can either be set to enabled, disabled or to act as a backup. In backup mode the interface is only used in the event no other network interface is available to be used. As MPTCP has matured, the path management functionality has been modularized, making management decisions flexible and extensible. The following list provides an overview of current path managers that are built into the MPTCP Linux implementation, as well as novel proposals for specific domains such as wireless environments and data centres:

- **ndiffports** – The ndiffports path manager is simple, as it creates a predetermined number of subflows between a single pair of IP addresses between the hosts. This path manager can be beneficial in a data centre environment, in which the hosts are single-homed but the network load balances TCP flows across redundant paths.

- **Full Mesh** – The full mesh path manager creates a TCP subflow for every pair of IP addresses present at the two end points. The remote host does not attempt to create any subflows, however, it does advertise additional

addresses that are present. The server advertises its available IP addresses to the client and for each advertised address the path manager creates a subflow for each of its own addresses. This path manager attempts to take advantage of both resource pooling and path diversity, to improve bandwidth and resilience.

- **MPTCP-MA** – In [165] the author focuses on using link quality information at the MAC layer to improve the decisions about how subflows are managed. Based on link quality estimation subflows are labelled either active or inactive. If the link quality is poor the subflow becomes inactive and its in-flight packets are re-transmitted using other available subflows.

- **A-MPTCP** – In [35], the authors propose that information from the user space can be used to inform the number of subflows that are created in a cloud environment. For example, if there are multiple redundant paths in a data center the user space application informs the MPTCP path manager of the available paths, leading to the optimal number of subflows being created.

2.2.4 Scheduling

In addition to path management, MPTCP also provides a modular structure to determine how packets should be scheduled over each path. The default approach is to schedule each packet to be sent on the path which has both the lowest round trip time (RTT) and available space in the congestion window. This lowest-RTT first approach provides an effective method of handling paths with heterogeneous network metrics. However, in the case that a flows throughput is limited by the application and not the network, this can lead to only a single subflow being used. Alternative scheduling schemes such as the round-robin scheduler, in which packets are allocated sequentially to all subflows equally, provides a more balanced approach allocating packets to all paths. This can be detrimental however, when the set of paths is diverse leading to low quality paths hampering the effective bandwidth.

2.2.5 Mobility

One of the implicit benefits of the MPTCP proposal is the inherent support for mobility [137]. As multiple subflows must be supported for a single connection end-point the identifiers and locaters that are used are split. This split allows new MPTCP subflows to associate with the initial connection regardless of whether or not there is an active subflow. This decoupling between the subflows and the connections provides a number of benefits on top of Mobile IP and comparable network layer solutions. Typically Mobile IP requires the physical or link layer connectivity to change before a handover can occur from one network interface to another, known as break-before-make. With MPTCP it is no longer necessary to wait for a connection to break, as data for the same end-point can be transmitted simultaneously over multiple paths; a make-before-break model can be established, providing truly seamless handover as packets are shifted to a new path before the old path fails. A full review of alternative mobility techniques is presented in Appendix A. Implementing and supporting alternative dedicated mobility models such as Mobile IP [126] in the presence of MPTCP introduces additional unnecessary redundancy; furthermore, it over complicates routing and introduces superfluous overhead.

2.2.6 Summary

In this section we have presented Multipath-TCP (MPTCP), focusing on the operation and design of the current standard. Furthermore, we detailed import factors that must be considered when account for multiple paths; such as, path management, shared congestion control, and the scheduling of packets. Finally we discussed the ability of MPTCP to seamlessly support mobility due to the decoupling between the application layer socket and the identify tuple of the individual subflows.

CHAPTER 3

Crowdsourcing Connectivity

In the previous chapter we presented a layered survey of resource pooling technologies followed by an in depth review of MPTCP. Leveraging the mobility properties that are inherent within MPTCP can provide an efficient and robust mobile connectivity model; furthermore, allowing users to share network resource can improve connectivity through increased multiplicity and diversity. Extending this model to account for cooperation between users can help to address a number of problems that a single device cannot. Allowing users and devices to cooperate and share their connectivity has been a point of interest for some time [59] and is currently transitioning into the mobile domain [13] [105] [92]. These connectivity models are attempting to alleviate some of the problems we described in Chapter 1, focusing on the inadequate availability and capacity of wireless links. Currently there is a wide range of research and commercial products focusing on these problems [181] [166] [13] [33], which make it evident that the problem domain is significant and of importance to both users and service providers. For users, there is a need to obtain as much bandwidth as possible at a reasonable price; while service providers must meet this demand in a cost effective manner allowing them to generate revenue. To this end, cellular data plans are becoming increasingly expensive for users [106], and service providers are struggling to meet their requirements. Recently the cost of the necessary capacity surpassed the expected revenue [164]. To this end, both researchers and service providers are looking for approaches to alleviate the congested mobile network. The congestion experienced in cellular and mobile networks, is a perfect candidate for the resource pooling principle to present a solution; such that congestion can be balanced, in a responsive way, dynamically moving traffic across a set of diverse

paths based on the available bandwidth. As discussed in Chapter 2, MPTCP can link the congestion control between subflows, allowing the host to move traffic away from more congested paths onto less congested paths. Therefore, combining resource pooling with cooperative mobile networks could allow multiple users to improve the utilisation of the available network resource and additionally help to alleviate congestion, by adaptively moving traffic to less congested paths. In this chapter we first look at current research solutions that allow and promote cooperation between users, sharing the available connectivity to improve resiliency and capacity, and alleviate load on the congested cellular networks. We then go on to present the set of requirements for a cooperative mobile network that we believe are imperative to improve the usability and QoS that users receive while mobile.

3.1 Multihomed Mobile Networks

One of the main problems with capacity sharing in mobile networks is the single homed approach to mobile connectivity that is ingrained in the network stacks of operating systems, typically only using a single default route at a time. Further multihomed support normally requires complex configuration and intervention by the user. To this end, supporting multihoming throughout a mobile network, becomes even more complex as detailed in [110]. In this section, we will focus on how Internet connectivity is appropriated and shared between a set of interconnected hosts, specifically looking at how the protocols enable the use of multiple Internet connections and determine which will be used.

3.1.1 Interior Routing

In this section, we will discuss the interior routing aspects that allow hosts in a mobile network to route between multiple gateways. One of the first discussions of multihoming in Ad-Hoc networks was presented in [49]. Focusing on the proactive Ad-Hoc protocols and more specifically Optimized Link State Routing (OLSR), the authors detail a multihoming solution, supporting NAT and Mobile IP based gateways. The authors begin by presenting the problem domain of utilising multiple gateways which is achieved through periodically changing the default route, noting that UDP performance suffered and TCP connections are broken, as expected. To address this well-known problem, the authors propose

IP-in-IP tunneling to the appropriate gateway, to ensure that a change in default route does not break the existing connections. However, the actual scheme for selecting the best gateway is left as future work. Alternatively, in [14] the authors, embrace the use of the default route, avoiding the overhead incurred by tunneling, which may be more appropriate for light weight applications such as VoIP. While the use of the default route provides a lighter approach, it presents a trade off against flexibility. Changing the default route or point of attachment can lead to connections becoming unroutable, which may have a more severe impact on the application than tunneling. Source routing is also a viable option as presented in [19], this method requires that the address for each hop between the Mobile Node and the Internet Gateway is added to the IP header, and is subsequently stripped as it is forwarded.

More recently in [16], the authors present a tagging based approach to forward packets to the appropriate Internet Gateway. In this scheme, each of the Internet Gateways advertises a unique identifier (tag), which can be inserted into the ToS field of an IP packet. When a tag is advertised the hosts forming the mesh must configure the appropriate routes to match the tag to the correct next hop, once this has been accomplished the routing process becomes straightforward. The lightweight tagging approach can easily allow per-flow routing decisions to be made, however the authors do not identify a complete solution to ensuring uniqueness of the tags that are advertised, which is of significant importance as the proposed ToS field is limited to six bits, preventing the use of potential global identifiers such as IP or MAC addresses.

3.1.2 Gateway Discovery

Gateway discovery is necessary functionality to facilitate communications between hosts in an Ad-Hoc network and external networks, such as the Internet. In an infrastructure network, the same functionality can be addressed by protocols such as DHCP. In an Ad-Hoc environment there are additional challenges, as multiple gateways may exist and connectivity is not always guaranteed. The appropriate gateway discovery approach realistically depends on the type of Ad-Hoc network that is being supported. To this end, there are effectively three base types of gateway discovery that can be employed, proactive, reactive, and hybrid. The proactive approach requires the gateway to periodically broadcast

an advertisement. The reactive approach requires that hosts make a request for Internet connectivity, with the appropriate gateway responding; while the hybrid approach, will typically combine both proactive and reactive. Hybrid approaches attempt to reduce the overhead of advertisements in large networks, by limiting the scope of the broadcast, requiring hosts that are further away to request access. A full implementation and analysis of these base approaches can be found in [144].

In [50], the authors propose a proactive approach, building a spanning tree to route packets from the Mobile Node to the Internet Gateway. The authors present two different methods to building the tree; including, a table driven and a route driven approach. In both approaches, the tree is formed by the Mobile Nodes in the Ad-Hoc network sending out broadcasts and essentially tracking and incrementing the hop-count. The broadcast originates from the Internet Gateway and contains the source address for hosts to use as a default route. When this broadcast is received, the host replaces the default route with its own source address, increases the hop count and re-broadcasts to the next hop.

In [153], the authors perform gateway discovery using a typical reactive approach, relying on the route request packets that are standard within Ad-hoc On-Demand Distance Vector (AODV). The route request is broadcast from the source host, and re-broadcast until it reaches the destination, which is a gateway in this case. The gateway then responds by unicasting a proxy route request back to the source host. The route discovery process is limited by a timer, so if no gateway is observed, the route request must be re-transmitted.

In [7], the authors take a standard proactive approach, using Gateway Advertisement Packets (GWADVS). Each GWADV is periodically broadcasted by each gateway, and contains information regarding the identity and properties of the gateway. This is then flooded through the network, as each host that receives the GWADV, rebroadcasts it. Optimisations to the base proactive approach are introduced by including randomized delay before rebroadcasting, reducing the number of potential collisions. Furthermore, a GWADV is only re-broadcast if it has not been seen before, or a more optimal route has been discovered.

In [149], a more advanced hybrid scheme is presented, which causes a proactive GWADV to be broadcast, when the gateway receives a reactive Gateway Discovery (GWDSC) message. The TTL of the GWADV is set to equal the number of hops it took for the GWDSC to reach the gateway, which can help to stop

the Ad-Hoc network from being periodically flooded, reducing overhead.

Finally in [104] an alternative hybrid scheme is proposed based on building bi-directional routes. To this end, the gateways in the network broadcast a "Root Announcement"; when a host in the Ad-Hoc network receives a Root Announcement, it sends a Route Request to the parent host (the host which delivered the Root Announcement), as per typical reactive approaches, which is then forwarded back towards the gateway. This process guarantees that a bi-directional route is built between each host and the associated gateway, establishing a tree topology. This process is repeated periodically as per proactive routing protocols, to maintain the tree and spread up to date information regarding the gateways connectivity.

3.1.3 Gateway Selection

In this section, we present a range of work which determines how to select from a set of available gateways in a Mobile Ad-Hoc Network. In [153], the authors present a reactive load balancing scheme for multihomed Ad-Hoc networks, aiming to reduce congestion across the available Internet gateways. To this end, hosts within the mobile network attempt to select the least congested Internet gateway. As the solution is reactive, the Mobile Node broadcasts a request for an Internet gateway when it is needed. As with AODV, this request is re-broadcasted until it reaches a host with knowledge of an Internet Gateway or the Internet Gateway itself, at which point a reply is sent back via unicast. As the network may be multihomed, the Mobile Node requesting Internet Connectivity can receive multiple replies. The first reply that the Mobile Node receives is selected as the gateway for the current connection; thus to balance load, the authors propose delaying the replies depending on the load exhibited at the Internet Gateway. The authors show that this model reduces packet loss and improves the load balancing capabilities in comparison to selecting gateways based on the hop-count. If all the gateways are significantly congested, the introduction of delay for selection becomes an unnecessary overhead. The authors identified that a signalling scheme between Internet Gateways is necessary to further optimize the protocol in this regard.

In [100] a system is presented for load balancing across multiple Internet Gateways in a MANET. The authors use a hybrid load balancing metric, to determine

the optimal Internet Gateway for a Mobile Node. This hybrid metric is a combination of the number of hops between the Mobile Node and the Internet Gateway, the number of Mobile Nodes registered to a specific Internet Gateway, and the intra-MANET traffic load. To calculate the metric, each of these components is simply weighted and summed, the Internet Gateway with the lowest value is then chosen.

In [7], the authors propose the use of Ad-Hoc Networks as a feasible backbone infrastructure and focus on the performance improvements necessary to realize this model. There Gateway Selection algorithm, takes into account both the quality of the available Internet Gateways as well as the routes required to reach them. This is broken down further into three metrics, the gateway load, route interference, and expected link quality. The gateway load is computed using the average queue length at the Internet Gateway. Route interference is measured as the combined interference of every physical link required to reach an Internet Gateway, interference can be determined by calculating the percentage of time that there is activity on the link, through carrier sensing. Finally the expected link quality is estimated by periodically broadcasting probe traffic, and determining quality by the success of the broadcast. Corroborating the approach presented in [100], the authors select the Internet Gateway with the lowest overall value, based on weighting and summing the discussed metrics.

To select a gateway in [149], the authors calculate the cost of each gateway using three factors, hop count, interface queue size, and number of neighbours. The hop count and interface queue size, have so far been typical metrics to use, the introduction of the number of neighbours offers a novel metric. By including the number of neighbours present at each hop, the density of a specific path can be taken into account. This can improve the internal and external network connectivity, as sparse paths are less likely to be congested leading to improved performance.

In [104], each host in the Ad-Hoc network selects a primary gateway, while the remaining gateways become a secondary backup. As [104] presents a proactive scheme for advertising gateway information, the period in between advertisements is used to measure load across each of the gateways and from the hosts. This information is subsequently used to allocate and reallocate gateways as either primary or secondary during the next advertisement. The gateways use the Knapsack Algorithm [108] (the parameters for which are: host traffic load, gate-

way traffic load, and number of hosts), to determine which set of hosts in the mesh should switch to a secondary gateway, creating a load balancing scheme.

3.2 User Cooperation

As discussed in the previous chapter, allowing users to cooperate and share connectivity has been proposed and investigated a number of times. As early as 2005, the Fonera Wireless Router was built, allowing users to share their bandwidth with other subscribers through an open and authenticated wireless hotspot [59]. The model for user cooperation is still an active area of interest and has seen numerous reiterations as the available wireless technologies evolve. To this end, user cooperation has a number of key benefits, collectively mobile devices can interconnect with the intention of optimizing a specific objective, such as minimizing power consumption for the mobile network as proposed in [77], or collaboratively downloading video content [94]. As discussed in Chapter 1 the ability for users to cooperate, share and interact is of significant importance to the future of mobile connectivity. For the remainder of this section, we present the current state of user cooperation for mobile networks, the models that are used for interconnecting the hosts and how Internet connectivity is ultimately shared. Finally we will present novel incentive approaches that are necessary for user cooperation to succeed and the associated benefits that can be obtained through this new connectivity model.

3.2.1 Cooperative Internet Connectivity

Offloading from cellular networks has become increasingly important as the cost of access rises with respect to the demand for mobile data. This has been shown to be viable in cooperative resource pooling protocols as shown in [176]. Similar schemes that enable user cooperation focus on accomplishing a specific task; for example, [94] focuses on improving video streaming by scheduling the download of video chunks between collaborating devices, based on the available bandwidth. Alternatively, [71] proposes the use of social networks to disseminate content between users, due to the negligible cost of sharing data locally, users become less reliant on expensive Internet connectivity.

COMBINE presents a solution enabling collaborative downloading in a mobile

network [5]. The proposal relies on WLAN connectivity between the hosts in the mobile network. To improve the energy efficiency, the authors introduce a new power saving mode for the wireless network interface, called the "Waiting Mode", which wakes up periodically to broadcast "I-am-Alive" messages. To split the collaborative download between users in the mobile network, the content is segmented and each chunk is added to a work queue; each of the hosts in the mobile network retrieve work from the queue, sequentially until it is complete. The authors present two options allowing COMBINE to download from unaware servers, the first is to either allocate a special-purpose proxy server, while the second is to assume that the server hosting the content supports HTTP byte-ranges, allowing each host to retrieve a predetermined segment.

In [162], the authors coordinate access between mobile devices, when each of the users in a mobile network are interested in streaming the same P2P video content. The hosts form a mobile network using a sharing link, which is assumed to be an Ad-Hoc WLAN connection or similar. Given the availability of the sharing link [163], the authors propose a decentralized approach to sharing connectivity information based on Multicast DNS [140]. Each host that joins the sharing link, sends a query for other hosts. Hosts already associated with the sharing link will reply, thus allowing them to form a connection. Subsequently, periodic checks are made to ensure that the connected hosts are still alive. Each of the hosts in the mobile network are assumed to have an access link that provides Internet connectivity. Content retrieved over the access link, can then be redistributed over the sharing link, offloading from the higher cost access links, such as cellular. In [163], the authors present two real world schedulers for coordinating the retrieval of chunks, a round robin approach and a historical approach. The round robin scheduler selects the next idle link to download the next chunk, which is similar to the work proposed in [5]. However, as [162] focuses on video streaming, chunks need to be downloaded before the point of playback; therefore, a round robin scheduler could lead to suboptimal performance as chunks are delayed on slower links. The second scheduler aims to reduce this problem, by keeping historical information regarding each links performance.

In [89], a bandwidth sharing mechanism is presented based on the concept of creating user profiles. These profiles contain policies describing how an individual user wants to collaborate, which are collated and processed using a Markov Decision Process [12]. To find other hosts, the authors propose the use of Bluetooth

discovery to obtain the set of MAC addresses that are in close proximity, this information is then uploaded to a server periodically (including host information), which makes decisions for the host, in terms of selecting helpers, whom can assist in the download process. The data shared between hosts is then transmitted over an Ad-Hoc WiFi connection. The authors define three policies to specify how cooperation between hosts can occur, an altruistic scheme, helper protection, and energy thresholding. In the altruistic scheme, a perfect world scenario is taken into account and assumes an entity has accurate state information with regards to each of the hosts. This policy allocates each download to the optimal host, even at the expense of the given hosts performance. The helper protection policy, aims to protect hosts from degrading their own performance when assisting others. The final policy uses an energy threshold; if the host has sufficient power to obtain its own content, it will use its own access link, otherwise it will request help from another host. The use of user profiles, can help to encourage users to participate in the system, by optimizing their chosen parameters, instead of making assumptions about what they determine to be the cost, thresholds for specific variables can be defined.

In MicroCast [94], the focus falls again on supporting cooperative video streaming. The authors present a scheme where a number of users wish to collaborate to watch the same piece of content. The MicroCast system is made up of three key components: MicroDownload, a scheduler that decides which user should download which piece of content; MicroNC-P2, which handles sharing of content locally, using overhearing and network coding; and MicroBroadcast, that enables broadcasts to be transmitted at a higher rate. The mobile network is formed using typical WLAN Ad-Hoc networking, under the assumption there are a small number of users, and that the users already trust one another. The scheduling algorithm attempts to maximise the use of cellular links especially under optimal conditions to ensure as much content can be downloaded as possible to help counter periods of disruption. Using overhearing in the MicroNC-P2 component helps to alleviate redundant data being transmitted over the local wireless link.

CrowdMAC [39] presents a crowdsourcing system for mobile access. The authors attempt to solve the following four problems: admission control, so a host relays sufficient data to be beneficial financially, but limits it such that network capabilities are not reduced; mobile access point selection, given a variety of hosts in close proximity, how does a host select the optimal gateway; how can hosts

attempt to solve challenges introduced by mobility when sharing connectivity; and finally how can the hosts address security and legality issues introduced through sharing. To allow the hosts to coordinate, while paying and billing for resources, a remote server is used, to aggregate and maintain user information. To determine access control, the Lyapunov framework [95] is used, which is an optimization model for dynamic systems that attempts to describe stability. The authors use the real network queue, a virtual queue representing projected load, and finally revenue, as the inputs for the model. To help handle mobility, the authors propose three approaches. Firstly, files are segmented to ease the downloading process, minimizing impact in the event that connectivity changes. Secondly, hosts are made aware of mobility and can choose access points that are more likely to remain available to them. Finally, the authors propose constraining mobility through incentive; such that, if the host providing connectivity breaks the connection mid segment, the cost paid for the service is returned, at the expense of the providing host. This could become a reason for hosts not to share their connectivity, a system that constrains and penalizes users from normal activity may not be seen favorably.

In [181], Incentive Data Plan Sharing System Based on Self-Organizing Network (INDAPSON) is presented, which present three categories of user: primary, assistant and detected. The primary users have a requirement to download content, the assistants are providing help to a primary user, and a detected user is a host whom may act as an assistant. The authors present two technologies that are required for the INDAPSON mobile network to form, Bluetooth management and a wireless transmission layer (802.11). The Bluetooth management stage is used during the initial formation of the mobile network, while the WLAN portion is used to exchange data. To collaboratively obtain a piece of content the authors propose a scheme similar to COMBINE [5]. When a primary user is provided with a URL, the content size is queried and split into segments which are then allocated to assistant users, until the entire piece of content has been retrieved; the segments are then shared over the WLAN link. Importantly, a host who has joined the INDAPSON mobile network, cannot become a member of a second mobile network, preventing the potential for complex, multi-hop topologies to form. During the formation and download process, a server is notified to keep a record of the assistance provided and data gained by each user. Assistant users are allocated based on a reputation scheme such that, the more a user helps oth-

ers the higher their reputation, and will therefore be more likely to have other users reciprocate.

The Collaborative Network Sharing (CoNes) architecture presented in [166], leverages both Software Defined Networking (SDN) and cloud services to provide inter-host connectivity and decision making processes. The authors present four problems that they wish to solve through the CoNes system: present a flat network abstraction over heterogeneity, fast network reconfigurations as host requirements and capabilities change, hosts in the network must be provided with a consistent view of the network, and active connections should not break during mobility events. To tackle these problems, the authors propose using Open vSwitch [131] to hide the heterogeneity of networking interfaces, enabling forwarding between hosts. To provide mobility a Virtual Private Network (VPN) is used, instead of typical standardised solutions such as Mobile IP [125]. During the formation of the mobile network, the hosts use multicast to discover one hop neighbours and subsequently measure the available throughput between them. This process is repeated periodically, with the throughput information being uploaded to a cloud service. The cloud service then responds, providing each host with a decision graph, describing how the available connectivity options should be used. The hosts then synchronize with one another via the cloud service, to acknowledge that the decision graph can be applied. The SDN forwarding model implemented, allows any host to act as a gateway, client, or relay at any moment in time. To ensure routability via other hosts, the sharing service on each host additionally performs Network Address Translation.

The use of Software Defined Networking for capacity sharing is proposed in [145]. The authors attempt to address a number of functional problems associated with traditional networking under the capacity sharing model. For example, a host in a mobile network is typically hidden from the gateway by one or more relay nodes. Therefore the source host is subject to the same traffic shaping, and rules as the host that is directly connected to the gateway. This presents Anonymity Authentication and Accounting (AAA) problems and potential legal issues depending on the nature of the traffic. To solve this problem, the Access Points that the mobile network connects to are assumed to be SDN enabled, allowing a host to check-in with the SDN controller, via the host which is acting as a gateway and the SDN AP. This effectively becomes an SDN based AAA approach for supporting mobile networks similar to the work proposed by [65].

This is a significant area of interest for the success of mobile networks, as users become more security conscious and providers have a need to ensure they are able to charge and identify users appropriately.

3.2.2 Modelling Incentive

One of the critical factors affecting cooperation between users is the ability to create incentive. If users do not see a benefit in sharing their connectivity with others, the entire system can fail, eradicating the network improvements that any cooperation system may bring.

One of the first proposals for user cooperation in Ad-Hoc network was presented in [20]. The scope of this work focuses on selfish hosts that prefer not to forward packets within the network. While the work focuses on internal Ad-Hoc connectivity, the work presents and tackles a founding concept for user cooperation. This concept is that helping another user, requires the expenditure of resource, which may be in the form of network, battery, or processing power. In [20], the authors aim to stimulate cooperation by introducing a simple counter at each host. The counter reflects whether or not a host is able to send its own packet. When forwarding a packet, the counter is increased by one, when sending a packet the counter is reduced by the estimated number of hops required. Thus, hosts that forward more packets, sharing their own resource are able to receive a better service. The simplicity of this model and the subsequent evaluation proves that encouraging users to cooperate can have a positive impact on the network, showing that decreasing the number of cooperative hosts decreases throughput. The feasibility of counting packets in current mobile networks is questionable, the authors identify this problem in terms of packet sizes being variable; however, not all hosts and links in a mobile network are equal and the cost or benefits for a given user to forward a packet can fluctuate. The problem of variable cost is considered in [36], which proposes a price based model, taking into account the power consumption and bandwidth usage to inform which route a packet should take. The pricing information is continuously updated and distributed throughout the network. The ability for a user to send a packet is represented as a monetary (credit) balance, which is increased when forwarding a packet and decreased when sending a packet, based on the associated cost for each action. The use of credits as a virtual currency has become one of the most common

ways for incentive to be modeled, ensuring that users are reimbursed fairly for the service they provide. In [15] the authors propose that the credit scheme can be presented as a game theoretic model, including both users, communities and service providers. The authors use the Stackelberg leadership model, which is based on the concept of leaders and followers. The community provider, i.e. the user sharing their connectivity, takes the role of the leader, while the Internet Service Provider (ISP) and users play the follower. The game is split into two levels, first the leader (community provider) determines how money is distributed, and secondly, the users and ISP's determine if they want to participate in the sharing community. Including the ISP in the incentive scheme is a novel approach that could still help to alleviate connectivity problems seen in mobile networks today, in which cellular providers limit or prevent sharing via tethering.

The work presented in [48], focuses on the problem of sharing WiFi connectivity in cities. The core of the proposal, revolves around a decentralized reciprocity scheme, using digital receipts authenticated against a public-private key pair. Individual WiFi communities are represented as clubs, and in order to gain membership to a club, the user must first contribute by sharing their connectivity with existing club members. To disseminate contribution and consumption in the form of digital receipts, the authors propose a gossiping algorithm, therefore when connecting to WiFi access points belonging to a club, the mobile devices will exchange the latest copies of their receipts. The metric to determine contribution and consumption rate only considers bytes transferred, which as previously mentioned may lack a realistic representation of cost to the user and more specifically may not represent the quality of the connectivity provided. In [84], a similar decentralized approach is presented in the context of autonomous networks. However, as opposed to gossiping and spreading consumption and contribution information, the authors propose a double auction algorithm, in which the members of the network bid, to buy and sell their network resource. The authors treat contribution and consumption as a Social Welfare Problem [96], the cost is raised for more selfish hosts that only attempt to maximise their own utility, while the reimbursement for a host that attempts to minimize the cost to itself is lowered. This model is then shown to maximise the social welfare problem, for a generalized network optimisation function, which changes depending on the type of network, such as Ad-Hoc or overlay.

A more recent research effort [182], uses incentive and sharing as a means

for hosts in a mobile network to conserve energy. The purpose of this work is to remove the complex economic models that we have previously discussed in favor of providing a simplified yet efficient interface. The authors propose that the economic models discussed can take a long time for currency of credit to establish worth, and predicting the real world implications and impact of such approaches is difficult. To circumvent these problems, when a host agrees to share its bandwidth the authors propose immediate compensation, whenever a Mobile Node successfully requests bandwidth, it will allocate a portion of its available bandwidth to the provider. Power savings are achieved through the bandwidth exchange process, as Mobile Nodes are shown to require a lower transmit power to forward to other hosts that are nearby. Even more recently [181] and [166] have promoted a centralized approach, relying on cloud services to support, authenticate and manage the incentive scheme. Both incentive schemes rely on the use of a virtual currency which allows users to effectively pay one another for bandwidth. [166] differs by using a game theoretic model, which is used to allocate the appropriate gateway for users in a mobile network, balancing QoS with both power consumption and financial cost, whereas [181] focuses on the reputation of the users to inform pricing decisions. Both of these new models provide a user-centric approach, putting the devices that are able to collaborate at the centre of the system. Alternatively in [64] [98], the authors propose an operator-assisted user-provided network. In this model, users are encouraged to share the operators network connectivity, in return for the operator providing them with a proportional amount of free data. By putting the operator in charge of both the benefit and cost aspect of sharing, a simpler model can be achieved, that users may be more inclined to support. The key problem with this approach is that it segments the users whom are able to share their connectivity, which can in turn reduce the diversity and heterogeneity, of the sharing model.

3.2.3 Opportunistic Networks

The world of Opportunistic Networking has evolved from MANET and Delay-Tolerant Networking, looking for novel approaches to efficiently support hosts through the use of mobile networks. This is a wide field ranging from technical network problems, such as the underlying infrastructure [22] [79], to application considerations such as caching within the network [28] [73], and determining high

level interactions between users using social models [72]. Opportunistic networking approaches need to solve three crucial problems as defined in [34]. Firstly, intermittent connectivity, as users and devices are mobile, the contact between hosts, and the lack of knowledge about location and direction and available connectivity options, can make communication challenging. Secondly, delay tolerant networking protocols must be leveraged and improved with novel caching and content management approaches, as users in the opportunistic network attempt to access remote resources. Finally the issue of heterogeneity, as devices, networks and services can be incredibly fragmented and diverse between users, providing appropriate mechanisms to abstract over or exploit the available diversity will be imperative to the success of the opportunistic model. Therefore there are a number of research challenges associated with opportunistic networks, presented in [34]. This includes areas such as, information management, context awareness, resource management, and economic and social cooperation. While the research in this area does not necessarily focus on leveraging and maximising the available external connectivity, the work is typically looking to solve similar problems as presented in Chapter 1. Such that, the ability to provide and benefit from a diverse set of connectivity options is an implicit part of the research area.

One of the key approaches proposed in recent years in this field, is the model of offloading from the congested cellular network onto an opportunistic network, which may have routes to cheaper connectivity options, or be able to provide the required content from a local cache. In a number of works, research has focused on how mobile networks can be created from online social networks, termed MoSoNets in [71]. By introducing the social paradigm into the mobile network space, incentive issues can be mitigated and shared interest can be exploited, under the assumption that social links can help to predict attraction to similar content.

In [27], the aim is to reduce the load on the cellular link. This can be achieved by disseminating shared content throughout a network or coordinating access to WiFi or Femto cells that may be a number of hops away. With the availability of additional heterogeneous links, it is possible to not only allocate links based on the application requirements but also based on the cost. This could relate to the users financial or power requirements for the device, in [179] mobile data is offloaded to a WiFi network in order to reduce the power consumption of the users device. In addition to user requirements, cellular networks are becoming increasingly

overloaded due to the significant growth of cloud services and multimedia on demand, making offloading even more vital. MADServer presented in [130] uses a Delay Tolerant Network (DTN) based architecture, which enables web content to be offloaded to opportunistic networks helping to alleviate congestion for cellular providers.

Context is of significant importance in an Opportunistic Network, and in [17], the authors propose three principal contexts. User context, service context and device context. In the user context, both personal and social information is of importance; for example, personal information includes, the users schedule, home address and work address, while social information presents links between users and places. Service context is based on the type of application that a user is currently running or is interested in, such as the available set of users or hosts whom can participate in the service. Finally, the device contexts are considered to be characteristics of the physical hardware, such as the available connectivity and battery capacity. All of these diverse contexts can be used to provide an opportunistic network with sufficient information to make optimal forwarding and data retrieval decisions. For example, the location and velocity of a set of users could be combined with available bandwidth, to determine if there is sufficient time to forward or share a piece of content.

3.3 Network Resource Management

The resource management domain encompasses a broad range of research. Since the emergence of multihoming in mobile devices, areas pertaining to the effective use of the additional network resource have become more prominent. In this section we will review the various approaches to the management of network resource in multihomed mobile devices.

3.3.1 Always Best Connected

The Always Best Connected (ABC) model for connectivity was one of the earliest concepts for management of multihoming in a mobile context. In [68], the authors present the concept of the ABC architecture and the associated challenges from both a user and business perspective. The base premise of ABC is based on exploiting diversity in access technologies and access providers to ensure that the

user is always connected to the optimal network.

The authors of [68] present a set of components that are required to support ABC, including: access discovery, access selection, AAA support, mobility management, profile handling, and content adaptation. While the definition of the components remains abstract the proposal and design is well grounded in addressing the challenges of the mobile domain and is of significant value. Access discovery requires a host to discover the available access networks, periodically updating this information to ensure there are no better alternatives. During the discovery stage, the authors propose that the available access networks provide the host with a set of parameters describing the quality and nature of the access network. Access selection presents both host based and network based approaches. The host based selection model, would allow the host to draw from user preferences to make a selection decision. A network based approach would take a broader, possibly centralised view; with a network service specifying the best access network for a host to use, which could help to alleviate congestion, by balancing users across different access networks. AAA support, is required to ensure both users and providers are protected, this is of significant importance if connecting via another user or roaming on an unknown network. Mobility management becomes even more important for mobile devices in the presence of ABC, as users may regularly switch between points of attachment as the state of network access changes, without appropriate mobility protocols, this would disrupt the users connections and degrade the quality of service; appropriate mobility models have been discussed in Appendix A. Profile management is presented in the context of an ABC provider hosting a users preferences, such that access networks are always aware of the users needs and requirements. Finally, the content adaptation component specifies that applications should be able to adapt to the current conditions. The adaptation may take form by providing applications with hooks into the ABC model; for example, a video may change the desired bitrate based on the current state of connectivity.

The work presented by Song et al. in [159] proposes a scheme to ensure users receive the best possible QoS at any given time. To this end, their solution is comprised of two components, they first evaluate the criteria and weightings of the QoS metrics, secondly the authors use grey relational analysis to rank the choices. This process attempts to balance the quality of the network against user preferences while also limiting frequent handoffs. Determining the most appropri-

ate network at any given time is a non trivial problem. As previously mentioned, the authors propose balancing network metrics with user preferences, which results in a multi-parameter or multi-objective optimisation problem. Therefore the use of well established mathematical models in such a scenario can help to simplify the problem domain.

Wilson et al. presented an alternative approach to optimise wireless access in [172]. Due to the complexity and uncertainty associated with choosing the best network, the authors propose the use of fuzzy logic to infer the optimal access network. This is justified in part, due to the subjective nature of potential QoS parameters; for example, users may be able to specify criteria such as "low latency" or "low cost", which makes the definition of an appropriate multi-parameter model very difficult. The use of fuzzy logic allows the authors to account for this ambiguity in requirements. As discussed by the authors, the core problem with this approach for network selection is that it requires a simple rule base, that must be built in advance. The addition of alternative QoS parameters may become challenging and overly complex, to counter these limitations the authors propose the use of learning algorithms, with the ultimate goal of matching the users behaviour.

3.3.2 Policy Based Network Management

In the realm of ABC, the authors typically discuss the use of "user profiles" or prior configuration on the end-host. Leveraging user policy to describe the approach to connectivity can help to minimise overly complex selection algorithms. A current example of such a tool is the Tasker [51] automation application for Android. Tasker simplifies the ABC model, by requiring the user to specify their network connectivity options based on a set of events and actions. For example, Tasker may react to battery capacity dropping and subsequently turn off the WiFi or Cellular interfaces. While Tasker is not built as a dedicated network management application, the ability to turn network interfaces on and off based on pre-defined events improves the basic connectivity model for mobile devices. In addition to simple applications, policy based network management has been a significant research problem in the core of networking, relating to service layer agreements, traffic management and ensuring QoS. Typically the policy based solutions focus on the core of the Internet and are of limited use at the edge,

leaving the end-host to make decisions for itself.

The user profiles specified in the ABC domain are typically simple; for example, prioritising access to one network over another. In [116], the authors propose putting the user and their associated context at the centre of the selection problem. To this end, the authors define a set of rules that can be used to limit energy consumption, for example, if their are no on-going communications turn off non-cellular interfaces, unless a user manually intervenes. By putting the users requirements at the core of the system, the multi-parameter optimisation problem can be addressed by a simple additive weighting model [1]. Additionally, the contexts the authors propose can increase the granularity of policy definition and improve network selection beyond looking at QoS. This is especially beneficial for applications that can adapt to the available bandwidth or network quality.

3.3.3 Path selection

The work presented in the ABC domain, has typically focused on the use of a single interface at a time. In the previous chapter, we introduced a range of resource pooling protocols that break this model, and rely on the simultaneous use of multiple network interfaces to maximise performance. To this end, path selection algorithms may provide a more fine grained approach to ABC. The general scope of the path selection problem is similar to ABC, as there is still a multi-parameter problem to solve, to define the best path.

In [93], the authors propose using active network measurements to help inform the path selection decision, sending probe traffic to gauge the RTT and the bottleneck bandwidth. This work focuses on addressing multihoming issues within SCTP, therefore each connection has a primary path and a set of backup paths. Given the set of paths and the associated metrics gathered by the probe traffic, the authors propose a simple set of rules to determine the most appropriate path. The rules comprise of a set of "if-statements", looking to determine which network path has the higher bandwidth, if both paths have sufficient bandwidth the round trip times are compared. As the solution is bandwidth driven, there is a possibility for a high rate of churn, as multiple SCTP connections may attempt to converge on the same path.

In [61], the focus is again on the multihoming properties associated with SCTP. The authors propose Wireless SCTP Extension (WiSE) to improve the

efficiency of an SCTP connection, by ensuring the best possible path is always used. Three extensions to SCTP are presented, focusing on: congestion control, path management, and bandwidth estimation. Firstly the authors attempt to distinguish between losses due to congestion on the path and losses due to bad channel conditions. This is achieved by comparing the current output rate of the connection to the last known estimation of bandwidth. If the output rate is higher, the authors assume the loss is due to congestion, while if it is lower the physical channel is held responsible. The path management approach is very simple, purely focusing on the bandwidth of each path. If an alternative path has a higher bandwidth than the current path, a switch will occur, resulting in the alternative path becoming the new primary path. Finally to estimate bandwidth the authors use both active and passive measurements, adding additional complexity in comparison to [93]. The primary path uses passive measurements due to the assumption that there will be guaranteed traffic, this simply compares the amount of data confirmed by the SACK and the RTT, to estimate bandwidth. As no traffic is sent on the backup paths, active network measurements are used, relying on sending a train of packet pairs as proposed in [76].

The work presented in [91] takes a QoS based approach to network selection closer to that presented in the ABC domain. The authors simplify the QoS optimisation problem by selecting a minimal set of parameters, including: monetary cost, area of coverage, required bandwidth, and the number of network interfaces to be connected. Furthermore, it is proposed that users be split into three categories: bronze, silver, and gold. Bronze users prioritize cost over QoS. Silver users desire a fair trade off between QoS and cost. While gold users demand the best QoS regardless of the cost. The authors then propose using an analytic hierarchy process to provide weightings for each QoS category for each user. Subsequently each path is provided an overall utility value based on the weight of criteria. The Euclidean distance is then used to determine which path is closest to the optimal values. The presented QoS metrics may not be sufficient in all cases to describe a users needs. As the authors focus on bandwidth and cost, important quality metrics such as loss, and delay are ignored. In the case of voice applications and other real time applications, high bandwidth is much less important than stability and delay. While there may be a clear relationship between loss, delay, and bandwidth; this can easily be broken for a number of reasons, such as access points and providers placing bandwidth limits on individual hosts.

3.4 Requirements

In this section we will reflect on the previously discussed work, and subsequently derive a set of requirements that are imperative for cooperative resource pooling to succeed in a mobile network. Essentially, all of the work discussed so far is limited in terms of resource pooling, despite the Internet connectivity that is made available. For instance, in [166] the authors proposed the use of Open vSwitch, masking multihoming from the network and transport layer. While this simplifies the required routing and puts the power in the hands of SDN, it limits the potential of bandwidth aggregation schemes that were discussed in the previous chapter; as protocols such as MPTCP typically rely on the network addresses being made available to the system.Therefore actively hiding or masking the multihomed properties from the network, transport or application layers could have a detrimental effect on both the hosts, and the networks ability to efficiently balance traffic and allocate flows.

In [166], the authors over simplify the mobility problem, only requiring the use of a VPN to tunnel traffic, which may lead to potential architectural issues. In a multi-hop environment, tunneling may become nested, taking a "ping-pong" routing approach when accessing the Internet. These issues may be exacerbated by increasing the complexity of the mobile network. The problem of nested tunneling has been of interest within the MANEMO community [110] and has resulted in complex signalling between home agents. To counter theses issues, we intend to use MPTCP to support mobility (with potential to support other transport or application layer mobility protocols). Therefore, network or link layer tunnels should only be used to provide additional security, and precautions should be taken when configuring routing rules to ensure nesting does not occur.

The existing work has shown that mobile networks and multihoming covers a diverse range of technologies, the majority of which are not compatible, requiring extensions or modifications to the current protocols to cooperate and work in harmony. To this end, we believe that the proposed solution should not be limited by technological boundaries; instead, an independent approach, running above the network layer would enable different link and network layer technologies to share connectivity seamlessly, bridging the gaps in what is typically a vastly heterogeneous environment. For example, if a user carrying multiple de-

vices, supported by a Personal Area Network (PAN) connects to an access point provided by a mesh backbone, the transition between the two networks can lead to multihomed connectivity being lost. By introducing the connectivity sharing at a higher layer, the complexities of proactive, reactive and hybrid routing protocols can be avoided when enabling intra-network communication. Despite this, an appropriate gateway discovery and selection scheme will still need to be designed and implemented. As the proposed solution will not be integrated with any current routing protocols, our design remains open and flexible beyond the work presented thus far in Section 3.1. Additionally, we do not intend to focus on a specific application domain as proposed in [94] or [162], we envisage any application being able to take advantage of cooperative resource pooling.

The role of incentive in a mobile network is a powerful concept and is imperative to the future success of cooperative networking. In the previous work [181] [166], there has been significant interest in this area modeling demand and sharing capabilities in a number of different ways. To this end, we do not envisage that incentive is an appropriate avenue of research, and instead current solutions to promote sharing should be implementable in the proposed cooperative resource pooling solution.

3.4.1 Terminology

For the remainder of this thesis, we will use specific terminology to provide a consistent and comprehensible syntax when discussing the different aspects of mobile networks and the associated connectivity, which are detailed below and depicted in Fig. 3.1 on the next page.

External Connectivity – In this thesis, the term external connectivity is used to represent a network interface that provides access to a remote IP capable host. This could include hosts that are connected to the Internet, or simply provide access to infrastructure that is not a part of the mobile network. With regards to a single host in a mobile network, there are two types of external connectivity that may be used, ***direct external connectivity*** and ***indirect external connectivity***, these types of external connectivity are not mutually exclusive. For example, external connectivity may be used both directly and indirectly, simultaneously.

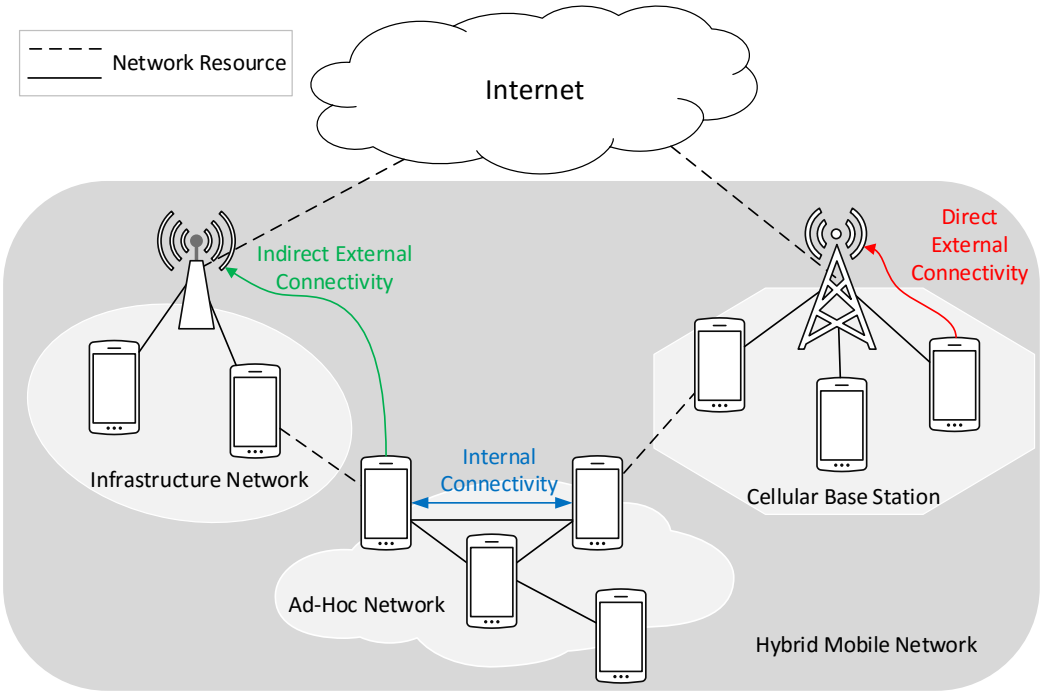


Figure 3.1: Example of a multihomed mobile network, with associated terminology.

Direct External Connectivity – A network interface that is physically attached to the host, providing external connectivity.
Indirect External Connectivity – A network interface physically attached to another host in the mobile network, that may be one or more hops away. Provides external connectivity to other hosts within the mobile network.

Network Resource – A network resource is a network interface that can be used to provide a host with a service. This could include, direct or indirect external connectivity, or alternatively internal connectivity that provides access to another host or service.

Hybrid Mobile Network – We consider a hybrid mobile network, to be a mobile network that consists of any combination of physical, link and network layer technologies. For example, a single host may support both ad-hoc and infrastructure WiFi access using multiple WLAN network interfaces, attaching to two distinct subnets. This host is then able to route between both the ad-hoc and the infrastructure networks, when necessary. Furthermore, the connecting host could interconnect using a diverse range of technologies, such as WiFi, Ethernet, Bluetooth or USB. There is no requirement for the connecting host to be a gateway, and may simply act as a relay between the two subnets. Finally, there may be one or more hosts connecting multiple subnets together, forming a single mobile network from a diverse range of mobile technologies.

3.4.2 System Requirements

This section contains the functional and quality requirements of the system, providing a detailed description of the specific features. The system is made up of two key components; a network protocol allowing hosts to announce and share connectivity, and a resource management system, allowing the available network resource to be managed and utilised based on the user, device and network context.

3.4.2.1 Multipath Advertisement Protocol

To achieve cooperative resource pooling, a network protocol is required that will advertise and share the available network resource, enabling multipath commu-

nications to be leveraged by each of the hosts in the network. This is defined as a network layer protocol, as we require a formal definition of the nature and type of external connectivity that is made available, such that it can be understood and processed by any compatible host.

Resource Monitoring – The underlying functionality requirement of the cooperative resource pooling approach is the ability to monitor changes in the hosts available network interfaces. Whenever a change in local connectivity is detected, this needs to be reflected in the applications current state. Furthermore, the different types of connectivity must be considered; for example, it is necessary for the software to distinguish between Internet connectivity and local connectivity. Finally the monitoring process should be dependent on a configuration file, enabling the user to include or exclude specific network resources in the network protocols scope.

Advertisement – Given the available network resource can be monitored efficiently, this information should be reflected by the network protocols advertisements, when new connectivity is established, or old connectivity is removed, the protocol should advertise this to other connected hosts. Furthermore, when an advertisement is received, the hosts are required to interpret and appropriately install the changes in connectivity information in to the applications state. As with the resource monitoring, the network resource that should accept or share network advertisements should be configurable, allowing users to only share or accept connectivity over specific interfaces.

Configuration – As the application is provided with local connectivity via resource monitoring and indirect connectivity via the advertisement process, the system will be required to configure the host accordingly, ensuring each connection is usable. Therefore, the available network connectivity whether direct or indirect, can be used seamlessly. Additionally, the network and routing configuration that is installed into the host, should be directly usable by any multihomed capable application or protocol.

3.4.2.2 User Policy Framework

The system shall be able to intelligently allocate and optimize the available network resource. This should not only optimize network performance but the QoS or QoE for the user, based on the current context. In this case, context may refer to the set of connectivity options, battery life, active applications, location, or any other variable relating to the state of the device or the user.

Context Framework – The context framework will form the basis of the resource management software. The core framework will not directly support any specific and implemented contexts, but will instead provide an interface such that any desired context can be plugged into the framework, in a modular fashion. The contexts to be considered, are network context and device context; examples of these could include: location, battery capacity, active applications, or network metrics. Due to the nature of the different types of context and how they should be handled, the design of the specific contexts will be split into two components within the framework. The contexts will require configuring, allowing the framework to monitor and address designated changes in context. These contexts should trigger actions or events when they meet or exceed pre-defined values. The action or event should then modify the hosts connectivity or application behaviour, depending on the users policy.

Policy Routing – Continuing from the configuration of the network protocol, the resource management software should be able to extend the routing rules using policy, to determine how specific applications or flows are routed. The policy routing will need to be informed by the previously discussed context framework to determine the appropriate policies to install. To this end, if the network protocol is active, the policy routing will need to query for how access to indirect connectivity should be routed. Furthermore, policies should be flexible, considering the type of traffic they are supporting; for example, applications or protocols may support soft handover, and thus may benefit from being migrated from one path to another. While other applications may suffer from such a process, the policy manager should, in this case, only change the path that new flows take.

Path Selection – The intention of the path selection algorithm is to allocate

individual applications or flows to the most appropriate path, as opposed to making coarse decisions based on load balancing, resource pooling or availability. To this end, the path selection algorithm should be aware of two key sets of metrics, the applications required performance and the current performance metrics of the available network interface. In keeping with the framework design, the path selection module should be pluggable at run time, and ultimately allow different path selection approaches to be used for different applications, services, or users.

3.4.3 Research Context and Methodology

In the previous section, the focus was on the objectives and requirements of the proposed system. For the remainder of this chapter, we will describe the methodology of the work, detailing how it will be carried out and more importantly, prove the proposed concept of cooperative resource pooling.

In [46], the authors propose three distinct paradigms for the field of computer science. The ***rationalist paradigm*** presents computer science as a branch of mathematics, treating programs as mathematical objects, in which a priori knowledge is valued. The ***technocratic paradigm*** treats computer science as an engineering discipline, in which a posteriori knowledge is sought after. Finally, the authors propose the ***scientific paradigm***, which presents computer science as an empirical or natural science, in which both a priori and posteriori knowledge are combined. Different paradigms may be increasingly valuable in different contexts; for example, the theoretical proof of coupled congestion control in [97] benefits from the rationalist paradigm, while the real world impact of MPTCP implementations as proposed in [124] is better proven through the technocratic paradigm. Due to the practical and pragmatic nature of this thesis, the technocratic paradigm will be employed, by designing and implementing a system to be evaluated, in both simulation and real world environments.

Combining both simulation and real world measurements increases the validity of the proposed system. The simulation approach to evaluation provides deterministic results, ensuring repeatability and the ability to tune variables in a controlled environment to understand how the system will react. Subsequently extending the evaluation and repeating in the real world improves the external validity of the experiments and can further justify the benefits and need for the proposed system. To further support our solution, the design and reliance on

protocols, should remain agnostic of the underlying hardware or software, providing generic guidance on how to achieve the intended benefits of the Multipath Advertisement Protocol and the User Policy Framework.

3.5 Summary

In this chapter we have presented a broad range of related concepts, from Mobile Ad-Hoc Networking to the opportunistic domain, and the ABC model. Taking these fields into account, we presented an overview of the proposed system and provided a detailed set of requirements that must be accounted for during the design and implementation stages. These requirements look to address problems identified in the mobile network domain by introducing resource pooling concepts, allowing network resource to be advertised and shared. Furthermore, we propose a flexible and extensible resource management framework, which adapts to both the users and the applications requirements in an effort to improve the QoS and QoE.

CHAPTER 4

DESIGN

This chapter presents the proposed design to enable efficient co-operative resource pooling throughout a mobile network, by disseminating the available external connectivity to all interested hosts. In the previous chapter, the need for a host centric approach to aggregating network resource was justified, discussing the limitations of current cooperative networking solutions. To this end, the Multipath Advertisement Protocol (MAP) has been proposed to enable hosts in a mobile network to take advantage of resource pooling technologies, to share and to aggregate network resource no matter where it resides. The chapter initially focuses on how to enable hosts to share and access direct and indirect network resource. Following on from this, specific design decisions are presented, enabling the support of complex network topologies. Finally, the User Policy Framework is introduced, based on the device and network contexts, breaking the typical resource pooling model that has been presented thus far. The aim of the User Policy Framework is to improve the QoS and QoE for the applications and users, which is achieved through fine grained resource management and path selection. This chapter draws heavily on previous peer reviewed work in [176], [177].

4.1 Overview

The aim of the MAP design is to allow the hosts in an already established mobile network to gain a complete topological view of the network resource that is available to them, and furthermore, make the network resource routable, which is not possible using current MANET or cooperative Internet access technologies. During the MAP configuration, the hosts in the network will advertise and

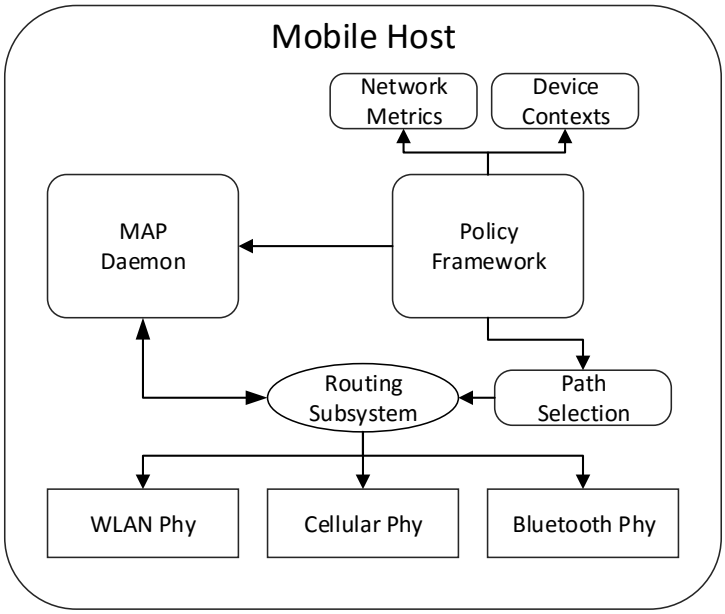


Figure 4.1: Design overview for a single host combining the MAP and the User Policy Framework.

disseminate the available Internet connectivity over the connected network interfaces. This process is similar to that of DHCP providing a host with a default route, however it is adapted to support multiple gateways that are distributed throughout the network, and additionally handle frequent changes in connectivity. The MAP process is further augmented with a User Policy Framework, which takes the routing information that has been obtained through resource sharing, and subsequently allocates a set of pre-defined applications to the most appropriate network resource in the mobile network. Furthermore, the User Policy Framework supports an adaptable policy mechanism which draws on the context of the device and the users preferences to intelligently control how the available network resource is used. The overview of the system components and the interaction with the hosts network and routing information is presented in Fig. 4.1.

4.1.1 System Example

In this section, the proposed system will be demonstrated end to end, this is presented through Fig. 4.3 and Fig. 4.4. In Fig. 4.3a we present the base network,

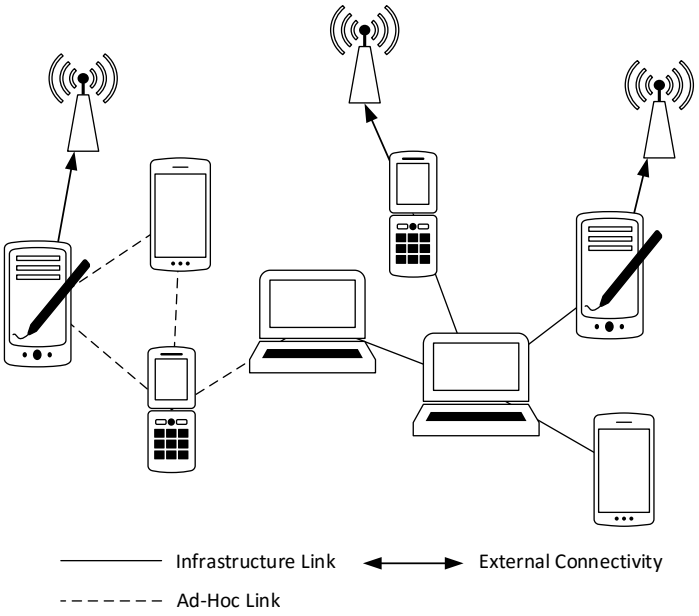


Figure 4.2: An example topology for a MAP enabled mobile network.

consisting of two hosts that wish to share their Internet connectivity. When the network connection is established, the MAP software residing on each host will make a request for the available connectivity in the form of a MAP request, as shown in Fig. 4.3b. When a host receives a request for connectivity, it will respond by transmitting the set of known external network connectivity, referred to as a MAP update demonstrated in Fig. 4.3c. In this case, this is a WAN and PPP connection for host H1, and a single PPP connection for host H2, along with the routing information to push packets from one hosts egress interface to another. Once the MAP updates have been received the routing and rules required to build the overlay are installed in each host as shown in Fig. 4.3d, this process effectively builds paths through to remote external connectivity, allowing it to be treated and allocated as direct Internet connectivity.

Once the routes have been established by the host, all the available connectivity in the network is free to be used. The User Policy Framework will first measure the quality of the available network connectivity, using active or passive measurements as shown in Fig. 4.4a. If the application is TCP based and can benefit from multipath communications, the default routing information created by the MAP process can allow an application to aggregate all of the available

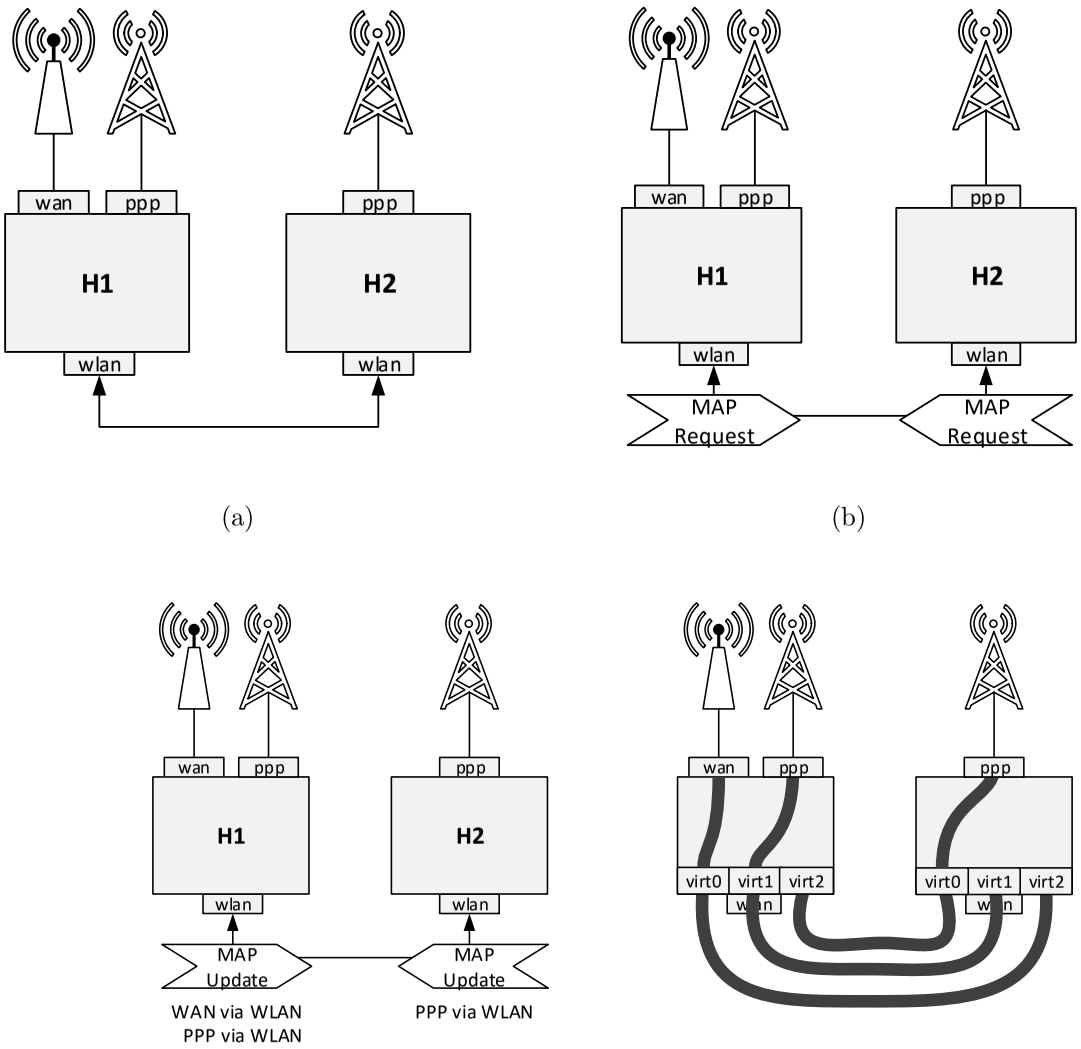


Figure 4.3: Worked example of MAP and the User Policy Framework

connectivity as shown in Fig. 4.4b. The aggregation approach chosen may include any of those presented in Chapter 2, such as MPTCP. Alternatively the measurement process carried out by the User Policy Framework allows the host to allocate specific applications to specific paths in the network, by creating and enforcing additional routing rules as shown in Fig. 4.4c. The User Policy Framework may also adapt the allocated connections based on the context of the device or network, as shown in Fig. 4.4c, an alternative path may be chosen based on changes in signal strength or battery capacity.

4.2 Multipath Advertisement Protocol

In order to provide a dynamic and flexible system, encompassing a wide variety of network topologies, it is important for the proposed solution to sit above any existing network configuration and routing protocols. The inter-connection of infrastructure and Ad-Hoc in complex mobile network topologies is common place. Therefore, limiting the proposed protocol to function within these arbitrary network boundaries may not lead to the most efficient usage of the available network connectivity. An example of a complex hybrid network is presented in Fig. 4.2 on page 77.

The proposed system relies on a host-centric information dissemination approach. Ensuring that each of the hosts in the network has as much knowledge as possible regarding the current context of the available Internet access. The context of the network has been determined to be a composition of the available resource, the associated metrics, current applications and inter-host mobility. As complex network topologies are to be supported, the proposed system needs to span a variety of different environments, formed from any combination of ad-hoc or infrastructure networks. This helps to justify the need for an independent protocol to disseminate the information throughout the network, as opposed to adapting current host configuration protocols, such as DHCP as proposed in [173]. When a link is brought up on a device, which may be due to mobility, pre-defined policies or user interaction, the information should be actively pushed into the network, notifying all other hosts of the newly available resource. Subsequently applications may decide to use the new link, if it is better suited to their requirements. Due to the vast increase in demand for data, it is also important that the

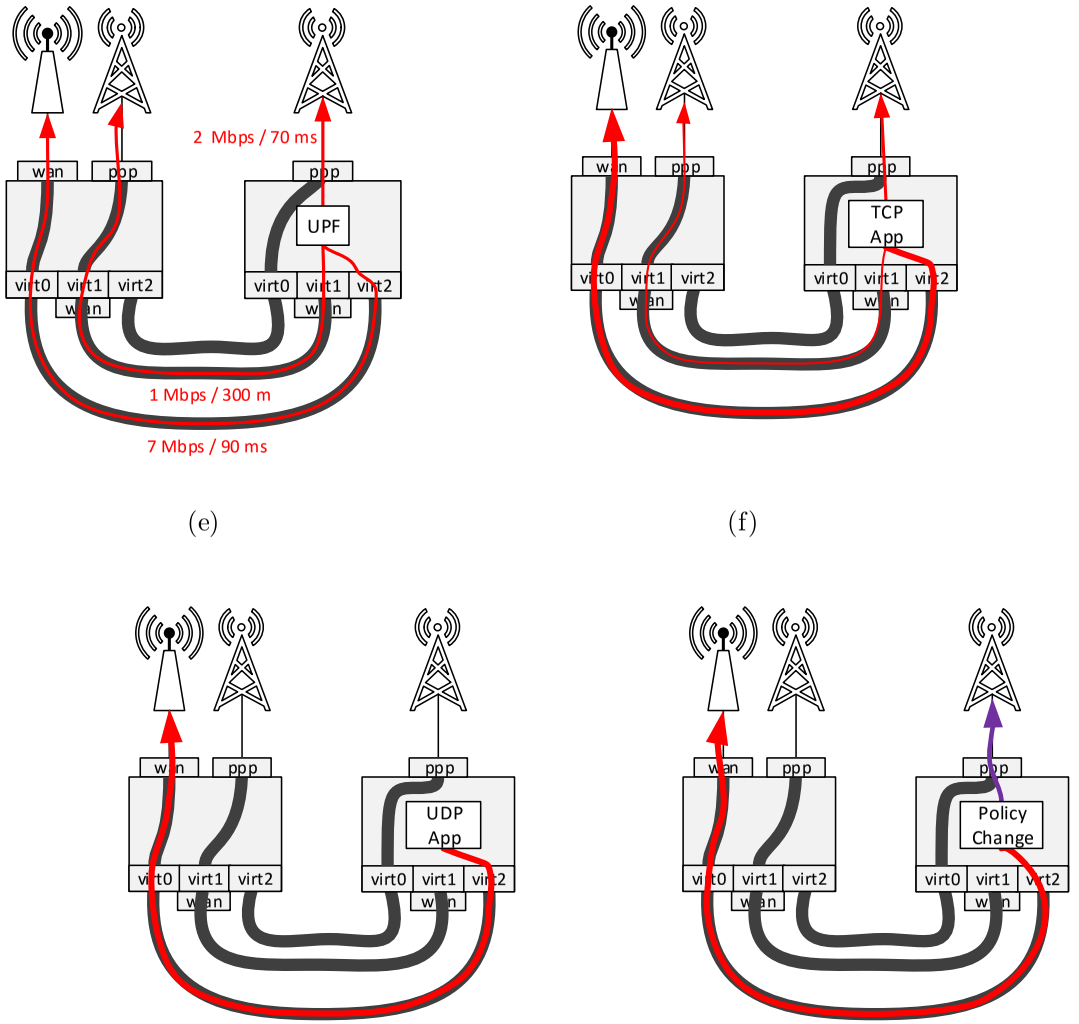


Figure 4.4: Worked example of MAP and the User Policy Framework

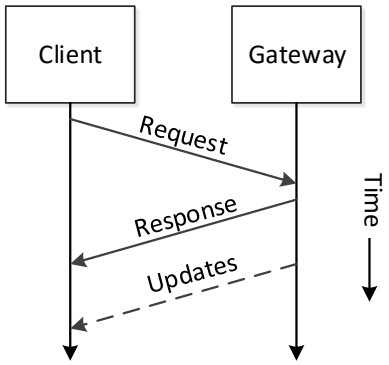


Figure 4.5: Message Exchange for the MAP.

proposed system supports the use of multipath protocols to enable the aggregation of the available network resource, which can help to facilitate high bandwidth applications, such as viewing high definition content and quickly accessing files from cloud storage providers.

4.2.1 Operation

The MAP is a connectionless network protocol, running above the transport layer. It employs UDP to advertise known Internet connectivity information to all hosts on the link. There are three key phases in the operation of the MAP, which are shown in Fig. 4.5. The three phases consist of requests, responses and updates. The process initiates with a client making a request for MAP updates on to each of the available network interfaces, the initial request packet can either be transmitted using broadcast or multicast.

The MAP initialisation takes place after the configuration of the network interface, if a host is given the IP address 192.168.1.10/24, in the case of broadcast the MAP request will be sent to 192.168.1.255/24. Responses to request packets and the continuous updates may be unicast, multicast or broadcast. Requests and continuous updates may be re-transmitted periodically depending on the reliability of the network. In the response and update messages, the protocol will provide routing information that will allow the host to create and establish its own routes. By allocating and installing the overlay routes after the initial addressing protocol has performed its role, we can allow hosts within the network to continue

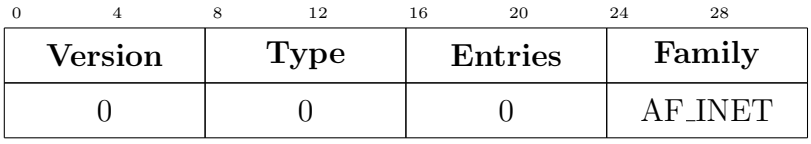


Figure 4.6: Representation of a MAP request packet in an IPv4 environment.

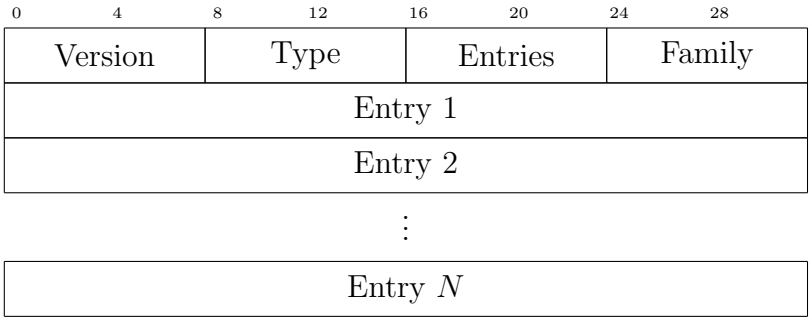


Figure 4.7: Representation of a MAP packet in an IPv4 environment.

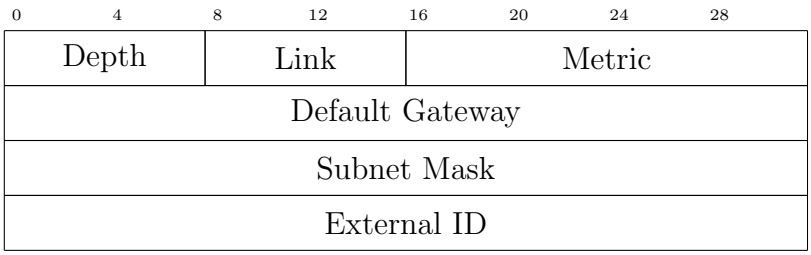


Figure 4.8: Representation of a network resource entry in the MAP, for IPv4.

communication as normal, while still allowing them to access additional remote resource seamlessly.

4.2.1.1 Header

The header of the base MAP packet is shown in Fig. 4.6. In total the header is four bytes long, containing four, one byte fields, that describe the protocol and the specific packet as follows:

Version – This field specifies the implementation version of the MAP in use, this is set to 0 for the reference implementation presented in this thesis.

Type – The type field specifies the kind of MAP packet, affecting how it should be processed. Type 0, specifies a request packet, while 1, 2 and 3 are used for update information. A value of 1, represents a full list of available interfaces, while 2 and 3 represents whether the entries listed should be added or removed respectively.

Entries – The entries field stores the number of network resources that are to be advertised or removed in this update. The field is unsigned and one byte in length, allowing for a total of 255 entries to be advertised in a mobile network.

Family – This specifies the address family for the contents of the packet, AF_INET for IPv4, or AF_INET6 for IPv6. An update can contain either IPv4 or IPv6 information, a mixture of the two is not permitted.

4.2.1.2 Requests

A MAP request packet indicates that there is a new MAP capable host in the networks subnet, all hosts in the broadcast domain that receive the packet will respond accordingly, as shown in Fig. 4.5 on page 81. The request packet only contains the MAP header (shown in Fig. 4.6 on the previous page), as no route information is necessary at this point in the capability exchange. If no response is received the request is periodically re-transmitted, the time between requests and the maximum number of requests is implementation dependent.

4.2.1.3 Updates

The packet shown in Fig. 4.7 on the previous page is made up of two major components, the header describing the contents of the packet, and the list of entries which detail how to create or update the hosts virtual interfaces. The entry defined in Fig. 4.8 on the previous page is as follows:

Depth – The depth field, represents the hop count between the host receiving the update and the host in possession of the remote network resource. If a remote network resource is a large number of hops away, the host may decide not to use it. This field can also be populated with alternative metrics that describe the quality

of the path between the host and the remote resource, which is to be specified by the implementation, alternative metrics could include signal strength, packet loss, bandwidth estimations, or queue sizes.

Link – The link value describes the physical medium in use to access Internet connectivity, to be more descriptive of the link type, this is split into two nibbles. The upper nibble describes the link type, while the lower nibble represents the category within the link type. For example, the link type may be Cellular, while the category could represent UMTS or HSPA.

Metric – The metric field describes the priority at the originating host. When the user or operating system initially configures the multihomed properties within the system, a metric value is assigned to each available Internet connection; the lower the metric value, the higher the priority of the link. For example, considering a multihomed laptop with access to Ethernet, WLAN, and Cellular, they could be assigned metric 1, 10 and 20 respectively, based on the quality of the backhaul. This metric value can be used to make simple decisions, determining how much traffic should be pushed over each link.

Default Gateway – The Default Gateway field specifies the IP address that the host should use to send its packets to, in order to reach the correct indirect network resource. For example, the Default Gateway provided may be 172.16.1.1, the recipient host will know it can then attempt to reach the Internet via this IP address.

Subnet Mask – The subnet mask is combined with the default gateway field to create the subnet that should be used by the receiving host. For example, given the 172.16.1.1 default gateway and a subnet mask of 255.255.255.0, the host will take its own primary identifier for the link (32 for example) and create the overlay address 172.16.1.32. This IP address is then used as the source for any packets that are transmitted and the destination for any packets that are received.

External ID – This represents a unique identifier for the network resource that is known globally throughout the mobile network. The core use of this identifier is to enable hosts to determine if two different network resources relate to the

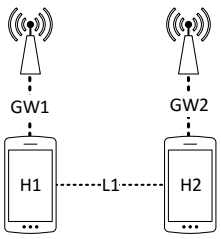


Figure 4.9: Two MAP enabled hosts connected via a single network connection.

same Internet connection, which is discussed in more detail in the loop avoidance and link backup sections.

4.2.2 MAP Behaviour

Thus far, the protocol described is able to support a client-gateway model, in which multihomed connectivity is disseminated downwards from a gateway to the hosts within the network. In this section we will present a number of modifications to the base protocol, which will be required in order to support more complex network topologies, and subsequently, improve the reliability and dependability of the network.

4.2.2.1 Loop Avoidance

The formation of loops in network topologies are a common occurrence, they can be useful in providing redundancy but at other layers are also identified to prevent packets from potentially looping infinitely. There are a number of feasible approaches to loop avoidance, that could be used by the MAP however as the overlay does not aim to enable host communication within the mobile network, each host only posses the minimum amount of information required to access the indirect external connectivity. Therefore no interior routing is made readily available, on top of what is pre-configured by the network. It is assumed that if the hosts are intended to communicate directly within the MAP domain, the routes will already be available to them.

Local Link Negation

In Fig. 4.9, a network topology is presented which may allow the formation of

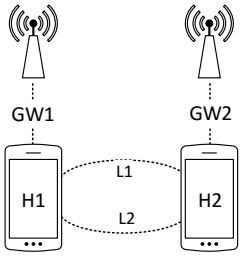


Figure 4.10: Two MAP enabled hosts connected via two network connections.

loops without explicit prevention measures being taken. In this case, both H1 and H2 have new Internet capable routes that they want to inform all other hosts about. Without preventative behaviour being implemented, the default would be for both H1 and H2 to push the new information back to each other, repeating indefinitely. The purpose of negating the local links, is to make the assumption that if a host receives an update on an interface, all other hosts connected directly or indirectly to that interface already know about or will obtain the information through another route. At the most basic level, this can be enforced by remembering which interfaces Internet connections were discovered on and then when it comes to pushing an update, the information is not relayed back on to the originating interface. In addition to preventing a loops forming in the network, by disallowing the replication of routing information back onto the originating interface this has the added benefit of reducing network overhead.

External Link Identification

When considering more complex network topologies, simply preventing the transmission of updates relating to the originating interface may not be sufficient to prevent loops from forming in the network. Considering the simplified topology shown in Fig. 4.10, each new link sent out over L1 from either H1 or H2 will be resent over L2 or vice versa. The simplest and most effective way to solve this problem, is to provide an external identifier for each advertised network resource as is shown in Fig. 4.8 on page 82. This field will allow the host to compare a received network resource to those that are currently known, if the external identifier matches a pre-existing network resource, it becomes apparent that this represents a different path through the network to the same resource. Determining the appropriate external identifier is left as an implementation decision. The

only design specification provided is the ability to uniquely identify an external network interface.

4.2.2.2 Stablility

If there is significant churn in the network, in which the available hosts or network resources are changing rapidly, there may be a detrimental effect on the hosts performance. For example, if a new network resource is announced and is incorporated into an MPTCP connection or is allocated for use by an application, the subsequent loss of the network resource, will lead to packet loss. In the case of MPTCP, this will lead to packets being re-transmitted over alternative paths, or in the case of a single path application, the connection will be broken and re-established over an alternative path. In this case, stability of paths should also be taken into account, to prevent paths with low availability from being used. To support this scenario, links should not be allocated immediately, and should instead wait for a pre-specified period of time before they are used. Alternatively opportunistic networking techniques may be used, for example, a GPS coordinate as well as direction of travel could be used when advertising network resource, to estimate if hosts are going to be connected for an extended period of time.

4.2.2.3 Link Backup List

In addition to preventing loops forming when disseminating Internet connectivity information, advertising the external link to be used has a secondary functionality. Adding the External Identifier, a host may be able to establish multiple different routes for the same egress interface, switching between them as internal links fail or become congested. The process of creating a backup list of routes to the same external interface could provide redundancy, reducing the handover time if the mobile network topology changes quickly. Previous research in multipath inside Ad-Hoc networks however has shown, that aggregating multiple TCP flows can be detrimental if the hosts are highly mobile [178]. Therefore, while this may be of future interest, it is not deemed necessary to further investigate the internal impact of MPTCP on throughput, in such an ad-hoc environment.

4.2.2.4 Subnet Collision Detection

If two independent MAP enabled networks connect via any given access medium, there is no approach available that is guaranteed to ensure there are no overlapping subnets, without an external entity regulating subnet provision. The potential for this to happen is mitigated by translating the addresses at each hop in the network, for this to be a problem, two directly connected hosts would have to share the same IP subnet, this problem could occur with either IPv4 or IPv6. The minimal behaviour specified by the MAP is for the hosts with the overlapping subnets to ignore the new matching subnet, preventing the dissemination of any potential Internet connections available on either side of the overlap. To resolve this issue, we present Subnet Collision Detection, at the simplest level this process requires one of the hosts with the overlapping subnet to choose a new subnet, and disseminate the change. For this change to work, the designated overlapping host must first query the directly connected hosts, to confirm that the new subnet is not in direct conflict with one of their subnets, at this point the designated host can push the new subnet back to the host it originally collided with. The query would take the form of an update message with a different ***type*** number than those presented so far, with the response again incrementing the ***type*** dependent on the query being successful or not. A change of IP address will undoubtedly have an impact on the active connections. This is effectively only problematic for hosts which are required to actively change one of their overlay IP addresses. For connections established by child nodes, i.e. nodes that do not have to change their overlay addresses, the presence of NAT at each host, will prevent their connection from breaking. For hosts changing a directly attached overlay IP address, this will break their connection, and the assumption is for the mobility protocol or application to handle the change. In the case of MPTCP, the connection will persist, creating a new subflow for the new IP address with the same connection identifier.

4.2.2.5 Accounting and Authentication

The proposed protocol provides a lightweight approach for hosts to share and use additional network resource. With the current set of information this will work well with users or hosts that are happy to share altruistically, which could match well to Use Case Two and Use Case Three, as presented in the introduction. In

Use Case One, however, the users and hosts may not know each other, therefore there is no pre-established or implicit trust between the devices in the mobile network. To solve this problem, we draw on work defined in Chapter 3, the protocol header should be extended to include an additional authentication field, created using asymmetric encryption. The authentication field should contain a privately signed key in both the request and update packets, the public key provided by a third-party management server can then be used to validate the user and subsequently provide accounting details to the server. The accounting interface should then inform the management server with information regarding usage levels. This will additionally provide the hosts and the MAP implementation to support access control and incentive mechanisms as discussed in Chapter 3. If both the provider and the client report different usage levels, this could be used to influence a reputation score similar to that proposed in [181]. Alternatively, to help enforce accurate accounting the SDN based approach proposed in [145], which allows the controller to provide AAA services could be used.

4.2.2.6 Security

A significant problem with any cooperative Internet access scheme, is the ability for users to maintain confidentiality and anonymity during the transmission of data. Any user that is relaying data for the end-host has the capacity to capture packets, or attempt a man-in-the-middle attack in an effort to hijack a users connections. While application layer and transport layer encryption such as SSL/TLS or TCPCrypt may appropriately secure the contents of any communication, relaying hosts will still be able to determine who the clients are communicating with, which may be of significant importance to some users. While a completely secure solution is considered to be out of scope for this thesis, it is assumed that users that are concerned with such an Internet access approach, would be able to securely tunnel their traffic to a trusted server in the Internet, before relaying it to the destination. As the proposed MAP design suggests the use of multiple network interfaces simultaneously, it would also be necessary to maintain a tunnel for each of the network resources that the host wishes to use, the configuration of which may prove to be challenging in such a dynamic and adaptive environment, as open and proprietary VPN solutions typically do not support multihoming to the degree that would be required. The associated

problems with maintaining and supporting multiple tunnels in this way are being met by the Mobile IP community, as they attempt to adapt the protocol to the multihomed domain.

4.2.2.7 MPTCP Integration

During the design and implementation of Multipath-TCP, the capacity sharing context was not considered. In the mobile domain, the focus was purely on providing a multipath transport layer from the perspective of a single multihomed host. Because of this, research to date has not investigated the impact that increasingly large numbers of subflows has on the host, or the total network utilisation. In [136], the maximum number considered was limited to eight; the authors discuss memory optimisations and evaluate the impact that increasing the number of subflows has on the host, in terms of CPU usage, memory consumption and achieved throughput. The current release of MPTCP supports up to a maximum of 32 subflows, due to the use of a 32 bit bitmask for subflow identification. While this may be more than sufficient for a single multihomed device, it is obvious to see that in a MAP enabled mobile network, 32 subflows could quickly be exhausted, especially in the case that the target server is multihomed. For example, assuming the full mesh path manager, if a host in a MAP enabled network establishes a TCP connection with a multihomed server; any more than 16 Internet connections in the MAP network would be redundant, limiting the potential utilisation of the available resource. Use of the full mesh path manager further exacerbates this issue, as the maximum number of subflows that can be associated with a single local or remote address is limited to eight. During the implementation of the complete MAP software and cooperative resource pooling system, we will provide a lightweight approach to increase the number of subflows that can be associated with a single connection.

4.2.3 Software Design

For the remainder of this section, we will present and discuss the proposed design for the MAP software, an architectural layout for the required components is presented in Fig. 4.11 on the next page. As the hosts view of the available connectivity will need to update based on its own network interfaces, as well as the Internet connectivity of other hosts in the network, there are two main

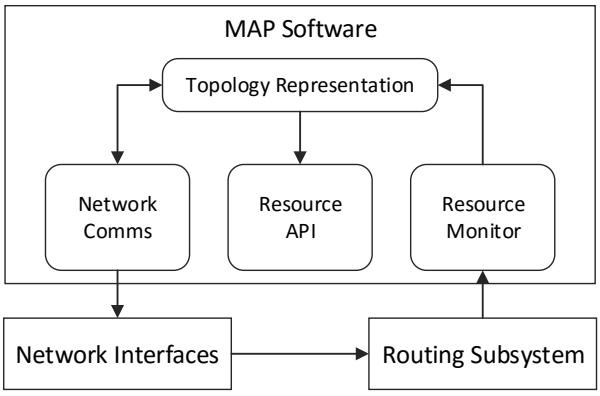


Figure 4.11: Proposed architecture for the MAP system.

sources of updates. Local updates and network updates, to monitor both of these processes will require a threaded or event driven model, allowing each to be monitored simultaneously. The base application logic for the MAP design is presented in Fig. 4.12 on the next page. This figure presents the two core paths and events that provide the foundation for the MAP software.

4.2.3.1 Resource Monitor

The resource monitor is the core component for handling direct network resource, that may or may not provide external connectivity. The resource monitor should ideally register with the routing subsystem to receive callbacks when, link state, network address, or route changes occur. If the state of the physical link changes, then this needs to be reflected even if the network interface does not lose its IP address, for example, an interface may be manually set to an "off" state by the user, at which point the MAP process should stop advertising and attempting to send packets via the network interface. The reverse is true if the network interface is set to "up". If an IP address is changed, added or removed, this will typically indicate a mobility event. If the IP address is changed or added, this indicates the host is attached to a new subnet and should send a MAP request; if enabled the host should also advertise its own connectivity onto the new subnet. Finally if a change in default route is identified, this infers that changes have occurred in direct Internet connectivity, which should be reflected by triggering an update packet to be sent onto the appropriate interfaces. The monitoring and update

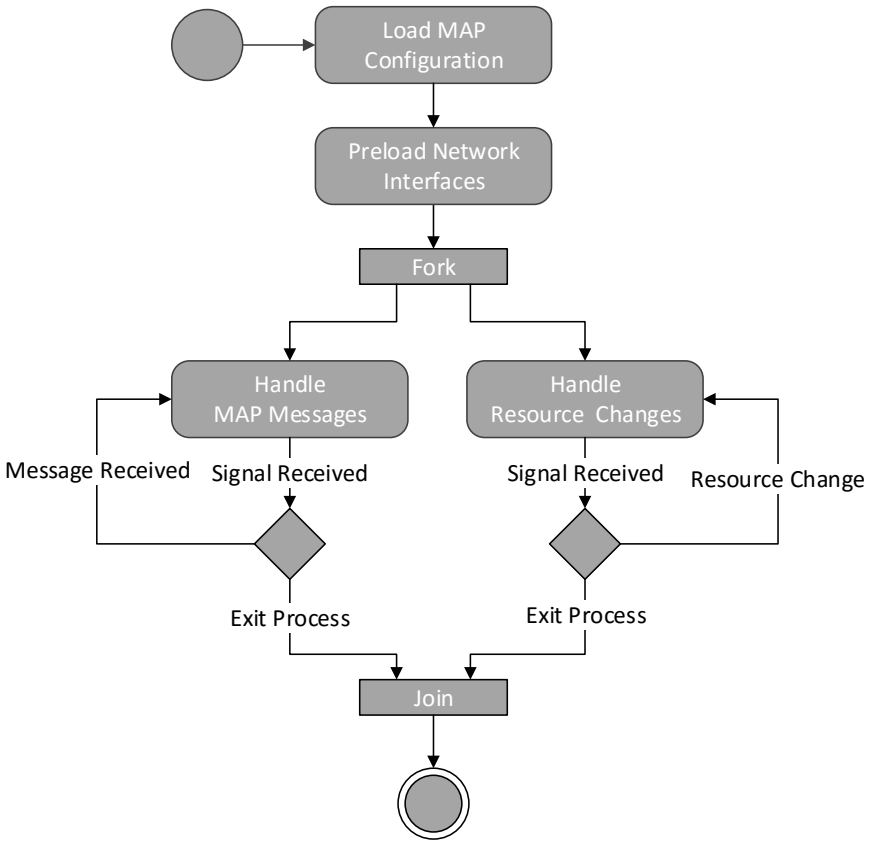


Figure 4.12: Application flow for the MAP software.

process that the resource monitor is required to perform is illustrated in Fig. 4.13 on the next page.

4.2.3.2 Network Interface

The network interface component is required to interact with other hosts in the network. As local updates are processed, they must be passed on to the network interface component, so they can be appropriately converted and serialized for the network, and deserialized on reception. The flow of the network interface component is presented in Fig. 4.14 on page 94, this activity diagram shows the sequence for receiving both requests and updates from the network. When an update is received, adding or removing Internet connectivity, the host must first update its local representation of the network topology and reconfigure its routes accordingly, subsequently, this new topology information must be re-broadcast

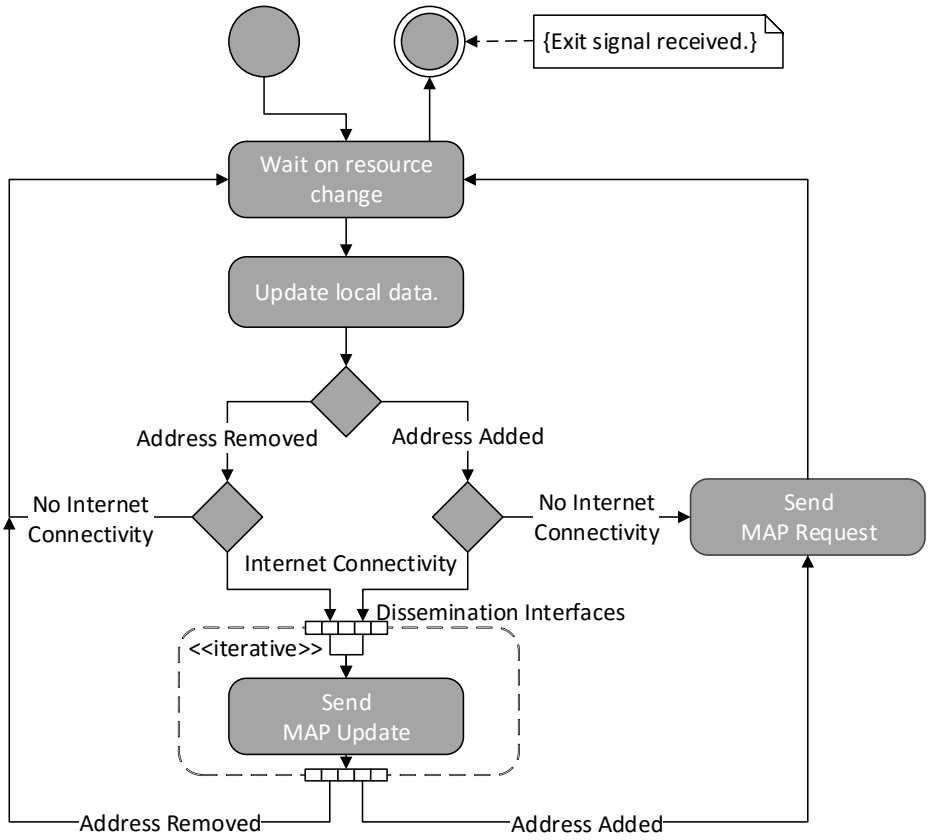


Figure 4.13: Application flow for the resource handling component of MAP.

to ensure all hosts in the network are updated. If a MAP request is received, the network component simply serializes the current network data representation and sends it back to the sending host.

4.2.3.3 Topology Representation

Internally within the software, it would be appropriate to maintain state regarding the available network resources. This will include, but is not limited to, the associated IP addresses and subnets, external identifiers, routing information, and whether or not they are capable of accessing the Internet via the network resource. This representation of the network topology will need to be updated by both the received network updates and any changes that take place in local connectivity. As we wish to treat all network resources as equal, it will be necessary to create an abstraction that represents network resource as a whole, with realisations that

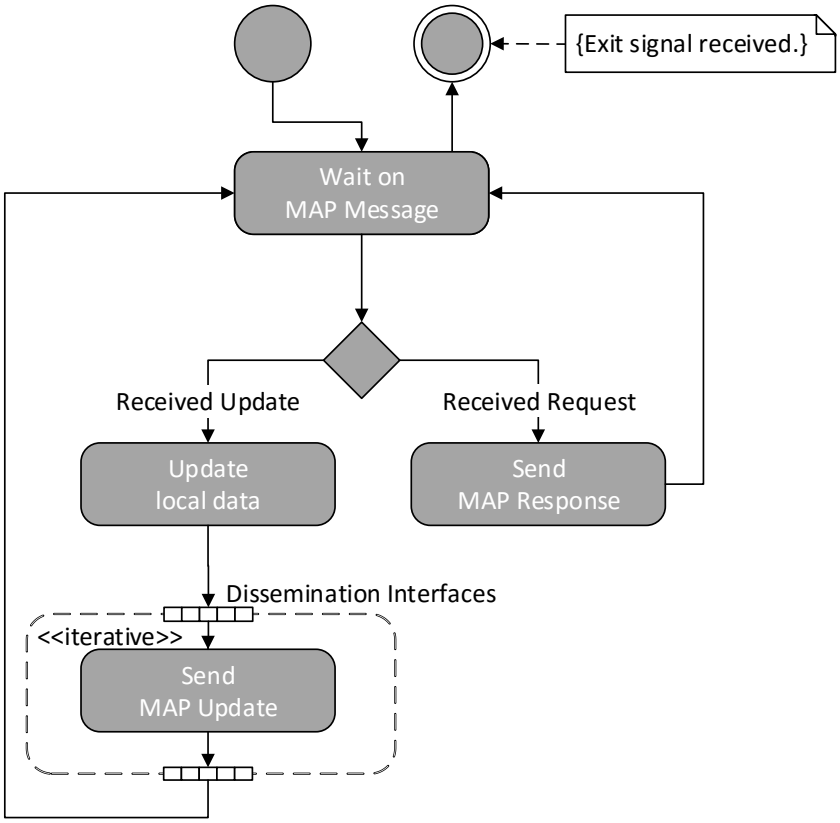


Figure 4.14: Application flow for the network component of MAP.

match to either the direct or indirect network resource.

4.2.3.4 Enforcing Routing

It is important to acknowledge that misuse and appropriation of network resource outside of the MAP advertisements definition is likely, whether malicious or accidental due to the mis-behaviour of applications. Therefore it will be beneficial for each host in the mobile network, to drop or ignore erroneous traffic. To ensure that a hosts network resource is not exploited, each relay should install firewall rules that block access to a specific network, aside from via the pre-determined route or subnet as detailed by the MAPs update packet. For example, if MAP only advertises the Internet connection ppp0, on the subnet wlan0, packets from wlan1 attempting to reach ppp0 should be dropped.

4.2.3.5 MAP API

Providing the host with a simple abstraction which describes the current state of network resource that has been discovered by MAP, will allow applications to benefit from the additional multiplicity and diversity; therefore creating more flexible and resilient network software. At the most basic level, the MAP API should be able to provide an interface which describes the available network resources and the Internet connectivity they provide. Furthermore, the API should provide a representation of how packets should be marked, tagged or addressed in order to be routed properly by the host to reach the destination. While the MAP software is not responsible for link quality information, each network resource should be updatable with a set of associated network metrics, this update process should be locked to ensure multiple processes do not attempt to write to the API during the same time period, as this could result in switching between different measurement approaches. This API can therefore be used by any application that wishes to support the use of multiple paths, without support from the underlying network or transport layers. For example, an application may create additional sockets, binding them to the network resource that is described by the MAP API. This could reduce the complexity required to build multipath capable applications as routing and interface management is already provided.

4.2.3.6 Configuration

Previously we have discussed the need for users to determine how their network resources are used. Therefore, the MAP software will need to be configurable. There are two key values that must be stored within the configuration, which includes the interfaces the user wishes to advertise on (where should updates be pushed), and the interfaces that shouldn't be disseminated by the software. For example, a host may not want to advertise the information for others to use their expensive cellular link, while they are happy for Ethernet or WiFi to be used openly.

4.3 User Policy Framework

So far in this chapter, we have focused on the base requirement for establishing a protocol to enable cooperative resource pooling within the mobile network. Given

the protocol implementation presented thus far, all available network resource is considered to be equal and shared fairly between the hosts in the network. The aim for this section is to define and justify the need to break this open access model for capacity sharing, and provide an appropriate design to achieve this goal. To do this, we propose the use of a User Policy Framework, which allows the user to specify how the available network connectivity is used, in addition to increasing access to network resource through the MAP.

In an isolated environment a single entity or individual may be responsible for all the available Internet access in a mobile network. One of the most prolific realisations of this model, is the usage of smart phones or tablets as a WiFi hotspot, providing Internet connectivity to the users other devices. This is one of the simplest use cases for the MAP, and the base cooperative resource pooling model is likely to provide sufficient support to meet the users needs. This model becomes more complex when additional users are introduced, as bandwidth limitations, power requirements and the economic cost of use may have a significant impact on the users decision to share their connectivity with others. The human factors associated with such sharing issues are considered to be out of scope for this thesis, however appropriate models in prior research should be applicable and implementable within the MAP, as discussed in Chapter 3 in [181] [166].

The purpose of the User Policy Framework is to provide fine grained control over access to the available network resource, and optimally allocate applications to the best available resource. The framework will be designed to be modular, extensible and flexible, allowing alternative or additional policies to be integrated easily. There are two key types of policy that we consider, context policies and application policies. Context based policies describe a set of factors in which a network resource may or may not be used, which are derived from the users device requirements; for example, do they want to maximise battery life, minimise cost, or aim for the highest possible QoE. Application policies dictate the optimal QoS required from the network for a specific application. The application policy and associated specification is used to assist in a path selection process, which attempts to match each defined application to the most appropriate network resource.

A partial taxonomy of different contexts is shown in Fig. 4.15 on the next page. The initial breakdown between types of context, splits between device and network context. A device context, represents measurable or quantifiable states

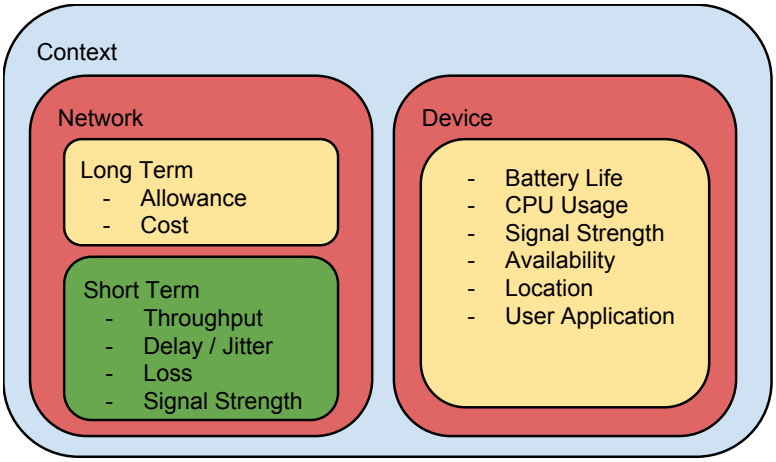


Figure 4.15: Taxonomy of different types of context for a mobile device.

for the device, such as the remaining battery capacity or the current CPU usage. The network context for a device, is split into two further categories, consisting of short term and long term contexts. Long term contexts may have an impact on the users decision to use specific network resource, such as the remaining monthly allowance for a cellular network interface. Short term contexts however, represent the current state of the available network resource, this includes the path metrics, such as available throughput and delay. The short term context, may also include lower layer measurements, such as signal strength. The decision process associated with the device context and long term network contexts, affects which network resources are available to be used, which will be referred to as the ***availability context***. The short term network context determines how the available network resource will be used, referred to as ***quality context***.

4.3.1 Context Policies

To provide a rich description of how a users network resource should be used we specify ***context policies***, allowing a user to establish precisely when a network resource can be used. When we discuss context we take this to mean the state of the device, the active applications and the available network connectivity. The current state of mobile connectivity policies are purely based on availability and are insufficient to fully describe the users needs. By specifying context policies for defining access to the available network resource, a users control over such decisions is improved. The criteria for managing these device policies, is based

on the current context of the device. By leveraging context, we can improve the current connectivity model, with regards to a users expectations of how network resource is accessed. Examples of contexts we can support include:

- **Battery Capacity** - Move traffic away from a cellular link when the devices power is too low. Or choose to use all available connectivity when the capacity is high to maximise the QoE.
- **Location** - Disable WiFi while on the move, re-enabling when arriving at known locations with stable connectivity. Or prevent the establishment of redundant short lived WiFi connections when moving at a high speed.
- **Bandwidth Allowances** - Limit traffic allocated to metered links after a certain percentage has been used. Alternatively, force WiFi to reconnect to an alternative access point when available and the bandwidth allowance is used up.
- **Network Quality** - Preemptively migrate traffic away from a link with a degrading signal quality or bandwidth.

This list is not exhaustive, and additional novel contexts can easily be included in the User Policy Framework due to our flexible policy definition approach. Furthermore any of these contexts could be concatenated to support more complex user requirements.

4.3.1.1 Configuration

To configure the use of such context policies, we propose a simple policy definition scheme, which determines the context that the user is interested in and how the device should react when the context is in a particular state. To avoid complex decision making processes in the implementation of the context parser, we consider that the nuances of conflicts between varying policies will be resolved manually by the user. For example, if one policy states to turn the WiFi interface off in a certain location, while another states it should be turned on in the same location, the expected behaviour is undefined. Policies however, should be additive allowing for complex configurations to be supported, such that a policy may state that a cellular interface is turned on, in a specific location, while another states it should be turned off in the same location, *provided the battery is*

```
policy := [conditions], [actions]
```

Listing 4.1: Syntax to define a policy.

```
condition := [key_id, link_id, value, comparator]
```

Listing 4.2: Syntax to define a condition to match.

low. This may be beneficial if the specified location is sufficiently far away from a point where the user can charge the device.

Policy Definition – Each context policy is described by the user in terms of conditions and actions. By specifying a set of ***conditions*** to be monitored and a set of ***actions*** to be performed when the conditions are met, network resources can be managed much more efficiently, as opposed to waiting for an interface to be handed over forcefully, or deferring to user intervention. The number of policies that can be defined is unlimited; additionally, the set of conditions and associated actions also have no hard limit as a prerequisite. The proposed policy definitions are declared as shown in Listing 4.1.

Conditions – Conditions are specified in terms of a context that we can monitor such as battery capacity reaching 50%. The conditions that a user is able to specify are highly configurable and the addition of new contexts or conditions simply requires a library that implements the context interface. The condition specification is shown in Listing 4.2.

The ***key_id*** is used to identify which context should be used to monitor for the condition being met. The ***value*** and ***comparator*** specify what state the measured context should be in for the condition to be triggered. The approach to reading and understanding conditions should be left to the appropriate implementation of a context, to ensure that the User Policy Framework does as little work as possible. Minimising the User Policy Frameworks involvement in the processing of conditions will improve the flexibility and extensibility when implementing additional contexts. Therefore, the User Policy Framework should not need to have any understanding regarding the condition that is being monitored, it should only need to know if the condition has been met or not.

```
1  action := [ip, do=[up, down, add, remove],
2             mode=[hard, soft], metadata]
```

Listing 4.3: Syntax describing an action to execute.

By describing the requirements of a policy as a set of conditions that must all be met, the policies become additive. For example, a policy to move traffic away from a local cellular link may have two conditions. The first condition specifies that the battery capacity has reduced to a pre-determined level, while the second condition states that an alternative link such as WiFi must be connected, before traffic is removed from the cellular interface. This approach ensures that a diverse and rich set of context and users requirements can be met.

Actions – The action specification is more concrete than the condition as the User Policy Framework is required to understand the content of the configuration, as shown in Listing 4.3. The ***ip*** identifies the network resource which the action must be performed on. The ***do*** parameter specifies the behaviour of the action, such as enabling or disabling access to a network resource. The ***mode*** parameter specifies the degree to which a network resource should be modified, if the parameter is set to hard, the system state of the network resource is changed. While setting the ***mode*** parameter to soft simply causes the behaviour within the User Policy Framework to change. This is a lighter approach that will respond to changes much faster than is required of the hard approach which causes the network interface to be configured and de-configured. The additional ***metadata*** is required when specifying addition and removal of MPTCP subflows while the interfaces remain active. For example, if the action specifies a particular application is migrated away from the cellular interface, the metadata parameter would require the identifying tuple for the associated subflows.

4.3.2 Application Policies

Thus far, we have focused on the benefits of resource pooling when a host is presented with multiple network resources. This focus doesn't account for the wide variety of applications that may not benefit from bandwidth aggregation, such as real time voice and video communications. No matter what the applica-

tion, it is still desirable to exploit path diversity to maximise the QoS that each application receives. Furthermore, there may be a subset of paths that are best suited for aggregation, preventing poor connections from decreasing the overall performance of an MPTCP connection.

Application policies define the QoS requirements that a user's high priority applications need to function in their optimal state. Application policies are specified in terms of the transport layer psuedo-header, any of which may be presented as a wild card. We then define the application requirements in terms of: bandwidth, latency, jitter and packet loss. To utilize these application descriptions the User Policy Framework is also responsible for monitoring the context of each network resource, obtaining the current network measurements for each resource. The combination of network resource measurements and application policy definition is then processed by a path selection algorithm to determine the best resource at any given time, a realisation of such an algorithm is presented in Section. 4.4.

4.3.2.1 Configuration

By configuring policies for specific applications it is possible for hosts to make routing decisions on a per flow basis. This provides much finer grained control of the available network resource than has typically been presented in the cooperative resource pooling domain, which allocates a host to a gateway. Optimal application decisions could be made entirely based on network metrics for each of the available resource, however, to further meet the demands of the user, it is also important to consider other factors such as energy consumption and economic cost. For example, a user may not be interested in using the absolute best network resource for background activity (such as periodic background updates), as this may incur significant cost, if the connection is metered, or the links data allowance is low. Where as, if the user wishes to make a VoIP call, then the QoE is much more important and the user may be willing to pay more for a better service. Specifying these criteria allows for a much more fine grained approach to allocating network resource than alternative solutions which may suggest allocating users to fixed price brackets. The work presented in [91], suggests that users either don't care about cost and just want the optimal service, or want the cheapest access possible. Enforcing such price brackets may not accurately

QoS Parameters	Description	Value
Bandwidth	The amount of bandwidth that can realistically be provided by the network resource.	Bits Per Second
Latency	Length of time taken for a packet to reach its destination (and/or return).	Time in milliseconds
Jitter	Observed variation in latency measurements.	Time in milliseconds
Packet Loss	Estimated number of packets that will not arrive.	Percentage
Economic Cost	Financially, how much will the user have to spend to use this link.	Pence Per Megabyte
Energy Cost	Estimated power consumption for the host device to use a given network resource.	Joules Per Megabyte
Availability	The uptime of the network resource, measuring reliability.	Time in seconds.

Table 4.1: Description of measurable network quality parameters.

represent all users; however, an optimal model could theoretically support these roles by minimizing or maximizing the importance of a cost based criteria.

When configuring application policies, it is important for the design to remain extensible as different device and application characteristics and measurements are considered. In Table 4.1 we present a base set of metrics that describe the benefit and cost of a particular link. Aside from economic cost, each of these parameters can be measured by the device, and further disseminated throughout the mobile network if necessary. In terms of the economic cost of a link, this could either be manually provided by the user, or retrieved automatically from a supporting network provider.

4.3.2.2 Network Measurements

In regard to obtaining network measurements there are two key approaches that can be taken, either passive or active. An active approach requires that a host injects probe or test traffic onto the network, subsequently monitoring the properties of the flow. For example, measuring round trip time can be achieved by sending out Internet Control Message Protocol (ICMP) echo requests and measuring the time taken to receive the response. Alternatively bandwidth could be actively measured by filling the path with TCP traffic to determine the number of

bytes that can be transmitted within a specific time interval. Gathering network metrics passively, requires the host to monitor and observe the characteristics of traffic that is already being transmitted. Bandwidth can be calculated by monitoring the amount of traffic that is sent and received, while round trip time could be determined by matching the timestamps of transmitted packets to the associated acknowledgements.

Measuring the quality of a network path both efficiently and accurately is still an open problem within the research community. While there are a number of active measurement tools that can provide reliable metrics, they can introduce significant overhead into the network to obtain accurate results. For example, tools like iPerf attempt to saturate the link to determine the available bandwidth. Alternative light weight estimation approaches have taken precedence in recent years as presented in [66]. Typically, these bandwidth estimation techniques aim to predict the available bandwidth by measuring the inter-arrival time for a train of packets with variable sizes.

In terms of collecting network measurements and other link metrics, there are two key areas to consider. Firstly, where in the network should the metrics be gathered, and secondly, which of the discussed active or passive approaches should be used.

Measurement Location

There are two feasible approaches that can be carried out in terms of collecting the network metrics for the path selection process. The measurements can be obtained, by either the directly connected host, or the host wishing to use the network resource. In the case of the directly connected host, the measurements would then need to be disseminated throughout the network. This could be accomplished via an extension to the MAP protocol, or an Remote Procedure Call (RPC) interface could be exposed, which allows a reactive query to be made, which is pushed up the network towards the gateway. From these two choices, we believe a RPC mechanism to be more appropriate, as extending MAP would unnecessarily increase overhead, pushing metric information to hosts that may not be interested. Alternatively for the host based solution, each host would continuously measure the quality of the available network resources, which in the case of active measurements could lead to a significant increase in overhead.

This would however, give each host a better representation of what the quality of the network resource looks like from their position in the network, additionally accounting for the interior path as well as the exterior Internet path. The potential improvement in accuracy from this approach could make it much more appealing, and could additionally provide faster feedback than including it in the dissemination approach.

Measurement Type

As the most appropriate type of network measurement may change depending on the connectivity in use and the location chosen, the proposed approach for the User Policy Framework is for the metric estimation to be implemented as a pluggable module, such that the appropriate measurement technique can be used on demand. A thorough evaluation of the advantages and disadvantages of active and passive measurements is considered out of scope for this thesis. Furthermore, measuring the quality of network resource in real time and subsequently migrating applications in real time to the most appropriate link could be detrimental, and lead to a negative feedback loop as all applications collectively move to the best link. Therefore, a hybrid approach combining both passive and active techniques could be of interest, only actively probing the network when there is insufficient information to determine quality passively.

4.3.2.3 Route Allocation

Given the path selection approach has allocated an application to a network resource, whether direct or indirect, the host must configure the appropriate routing decisions to ensure the application uses that path. For applications that support mobility, this process can be dynamic, actively migrating the traffic to a better link. If an application or protocol does not support mobility, the framework should only redirect new flows to use the new path, to avoid actively breaking the users current applications. As the MAP provides and maintains the appropriate routes to reach each network resource, route allocation for each application can be installed on a per host basis requiring no additional signalling within the network. This is advantageous as updating and maintaining these routes across the network for each device would incur a significant overhead, and ensuring consistency between the hosts would create additional problems, likely requiring

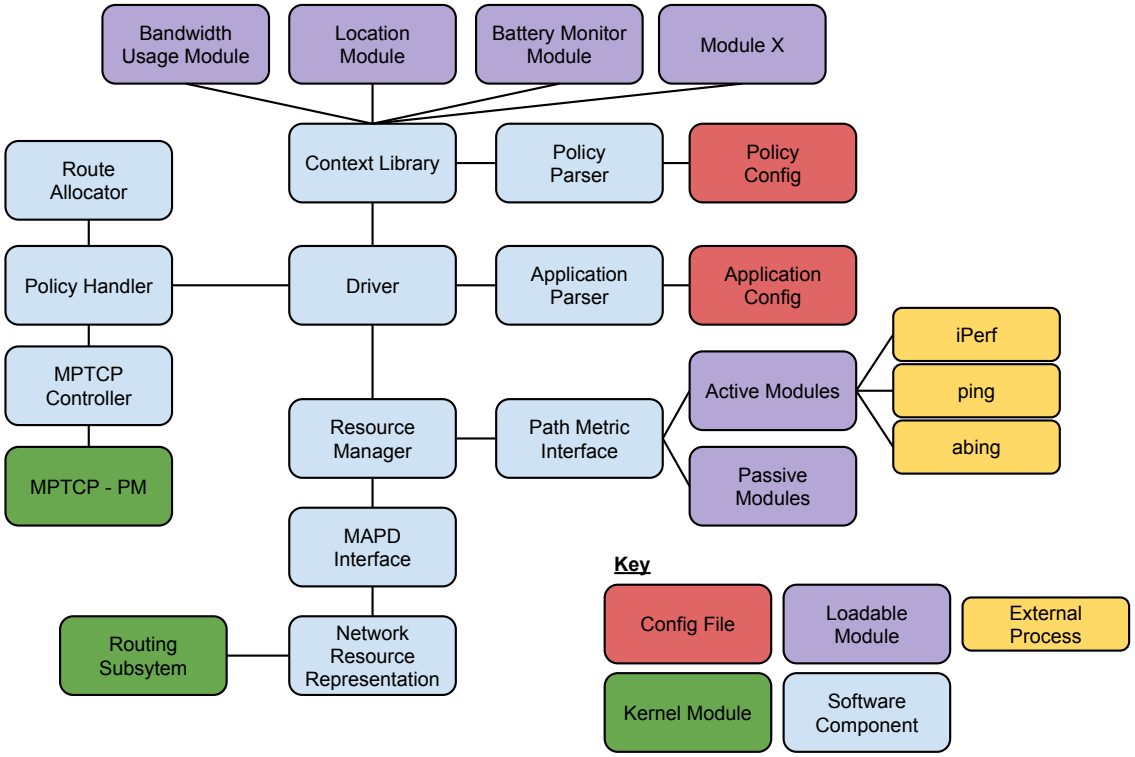


Figure 4.16: The architectural design for the User Policy Framework.

an external entity to regulate the mobile network.

4.3.3 Framework Design

The overall design for the User Policy Framework is presented in Fig. 4.16. To ensure the extensibility of the framework design, the proposed software should take an event driven approach, reacting to callbacks from the modules that are built. While the User Policy Framework should be able to learn about network resource and routing decisions from the MAP API, the framework is also independently a significant contribution, as context and policy routing at the edge for multihomed devices is still an open-ended problem. Therefore, the proposed design for the User Policy Framework should work in the same way, either on its own or with the MAP software running along side it. This means that if the MAP software is not running, the User Policy Framework should be able to obtain network resource information through its own component, replacing the MAP interface. This component can be significantly simpler than is needed by the MAP, as it only needs to monitor for the addition and deletion of default

routes, representing the available Internet connectivity, as opposed to link and network layer addressing.

In this section we present an overview of the User Policy Framework design, addressing the integration with MPTCP and subsequently the management of both context and application policies.

4.3.3.1 MPTCP Integration

In Chapter 2 we provided a detailed description of MPTCP and the available path managers and configuration that can take place to manage how paths are created and used. The default path manager builds a full mesh of IP addresses between the hosts, for each available TCP connection. This model is being augmented as the MPTCP community introduce multipath socket options for the application, allowing for more fine grained control. This still however, limits the MPTCP connection to being turned on or off, and does not allow for the flexibility desired by the User Policy Framework. To this end, it will be necessary to integrate the MPTCP implementation with the proposed User Policy Framework, and subsequently provide an interface which allows the users policy based decisions to have a direct impact on how flows are created and managed by the MPTCP path manager. In the framework design, the route enforcement and policy component is signalled by changes in the system and application state, which can lead to changes in path selection decisions. Therefore the policy routing should also be able to interact with MPTCP determining how the connection should behave. Therefore, the MPTCP path manager will need to communicate directly with the User Policy Framework, to both send and receive information regarding the current state of the active MPTCP connections. There are four functionality requirements that the proposed MPTCP path manager and User Policy Framework combination must support:

1. Notify MPTCP connection established.
2. Notify MPTCP connection destroyed.
3. Receive create subflow.
4. Receive destroy subflow.

This functionality will allow complete control over the available MPTCP connections, and furthermore will allow for changes in device and application context to initiate soft handover.

For the purpose of this framework, we introduce a new naming convention for multipath hosts and applications. Firstly, we assume all hosts are capable of supporting MPTCP, however not all applications may find a multipath connection desirable, short lived connections or delay sensitive applications may prefer to use a single path. To this end, specific applications may be either ***Multipath Enabled***, ***Multipath Capable***, or ***Multipath Disabled***. In the case of a ***Multipath Enabled*** application, the framework will always attempt to use the complete set of network resources. While a ***Multipath Capable*** application will typically prefer a single path approach, but it is capable of MPTCP functionality, such that the connection can be migrated when necessary, to either a better path, or to preempt or react to a path being broken. Finally if the flow is ***Multipath Disabled***, the framework will at no point attempt to leverage multipath functionality.

4.3.3.2 Callback Events

As previously discussed the User Policy Framework needs to incorporate a series of callbacks to support and provide the proposed modular structure. Based on the context and application policies that are defined, the presented framework requires four callbacks that must be serviced appropriately. These callbacks consist of:

1. A condition is met for a context policy.
2. The state of a network resource changes.
3. Network metrics change significantly.
4. A new MPTCP connection is established.

Condition Callback

The condition callback is presented in Fig. 4.17 on the next page. As discussed in the previous section, when a set of conditions are met, the action that is

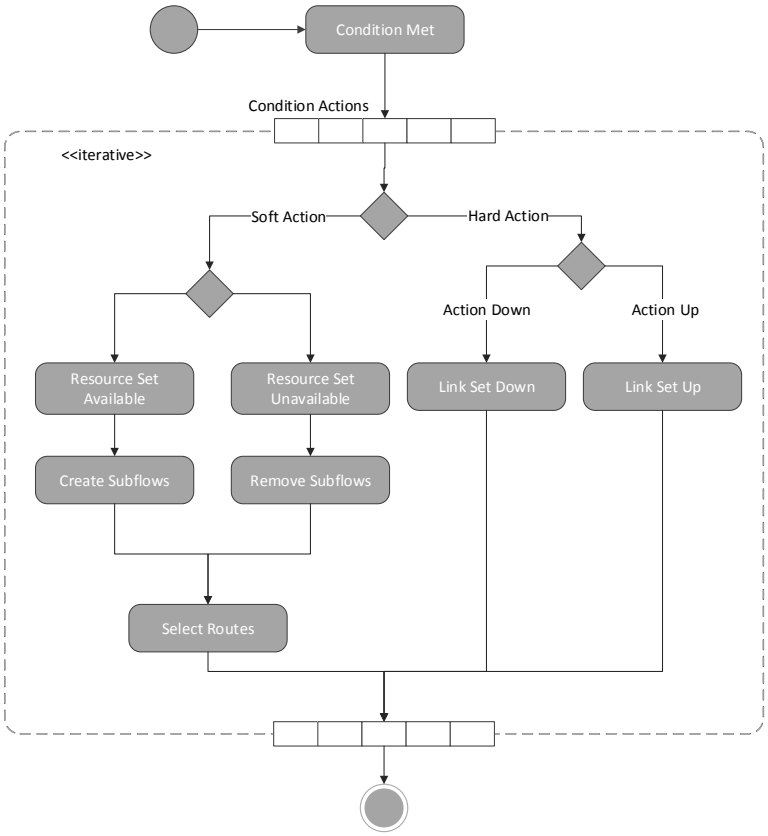


Figure 4.17: Application flow for meeting conditions.

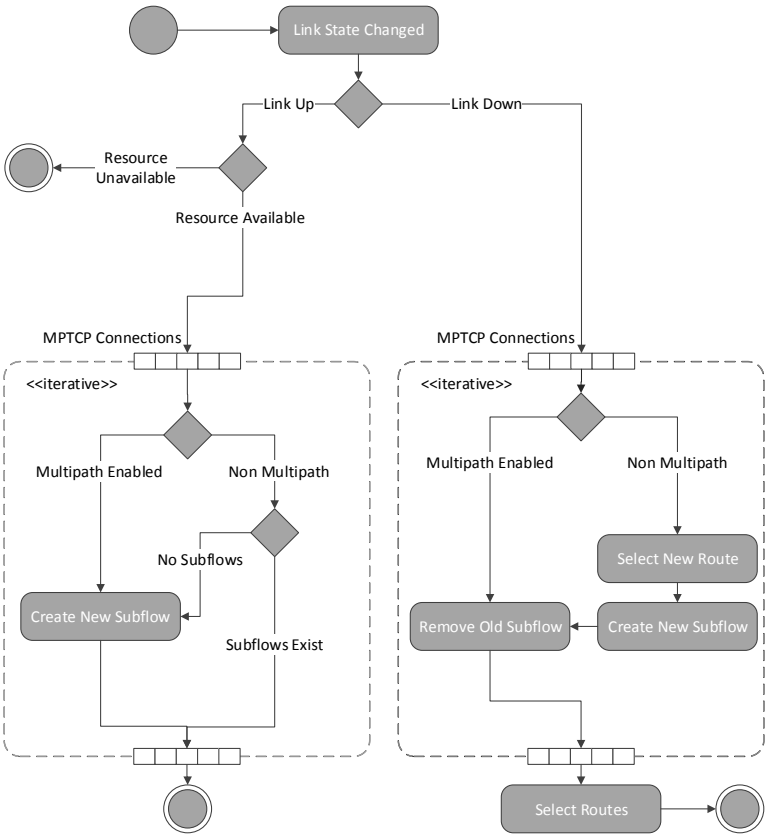


Figure 4.18: Application flow when the system link state changes.

carried out can either be hard or soft. The hard action represents a link that is being altered at the system level, such as deconfiguring the interface so that it cannot be used, while the soft action simply takes control of the interface within the software. A soft action may be useful for a WiFi interface that is actively scanning for a new AP that is usable, but the previous AP hit a usage limit, while a hard action may be more useful for a cellular interface to minimise power consumption by permanently putting it to sleep. A hard action will subsequently trigger a link state change, which will be handled by the resource monitor, to avoid duplicating functionality. The soft action however requires that current application paths, specifically MPTCP connections are modified to stop or start using the interface specified by the action. Subsequently, to finalize the condition callback, a new set of paths are selected based on the new availability information.

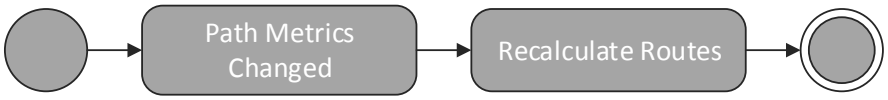


Figure 4.19: Application flow for a significant change in path metrics.

Resource Change Callback

When the availability of the set of network resource changes, multipath capable traffic must react accordingly. This model is presented in the activity diagram of Fig. 4.18 on the previous page. This callback could be triggered by changes to either direct or indirect Internet connectivity. In the event that a link goes down, traffic that is multipath enabled, should simply remove the old subflow as the complete set of potential subflows will already have been created. For multipath capable flows the path selection algorithm must first choose the next best path, and create a new subflow using this network resource, before destroying the old connection. In both cases, the path selection algorithm must be subsequently run for each application specification such that new flows will be routed over the newly chosen network resource.

If a new network resource becomes available, multipath enabled applications will create a new subflow utilising the new network resource, while multipath capable flows, will only create a new subflow if there are currently no active subflows. This effectively presents a hard, break-before-make handover, which is the worst case scenario in which there was no usable network resource available for a period of time. A new network resource becoming available will not trigger paths to be re-selected, as it must first wait on the network metrics being obtained or measured.

Metric Change Callback

As the path metric interface is continuously monitoring the current state of the retrieved network metrics, if there is any significant change detected, the path selection process will be triggered, recalculating the best routes for each application, shown in Fig. 4.19. Determining the magnitude of the change is important as if there are two similar network resources, it could force re-calculation of the

routes constantly, consuming unnecessary resource. Furthermore, if applications are being migrated as the path selection decisions change, this could lead to degraded performance as the path is altered too frequently. The approach for determining significance is left to the implementation. However there are a number of design choices to consider. The significance could depend on a single metric changing, or a composite of all metrics. Furthermore measuring the significance or magnitude could present a logical problem, as a percentage change in each network parameter will have a different impact on the perceived QoS or QoE. As discussed in Section 4.2.2.2, stability is an important factor for maintaining the best possible performance, oscillating between different paths can have a negative impact on the perceived quality of the network resource. To this end, when determining a change in network metrics, availability should also be used as a factor. Accounting for link availability will help to ensure that more reliable paths are favoured over ephemeral network resources, which can help in reducing packet loss, minimising the negative impact on QoS and QoE.

MPTCP Callback

The MPTCP callback is straightforward, as shown in Fig. 4.20 on the next page. This requires that the MPTCP path manager notifies the User Policy Framework of any new MPTCP connections that are created or destroyed. When a connection is created, and it matches a multipath enabled application, the policy manager initiates the creation of a subflow for each of the available network resources. The connection is also added to an internal representation of available MPTCP connections which are used in the previously presented activity diagrams in Figs. 4.17, 4.18, and 4.19. When an MPTCP connection is closed, the User Policy Framework simply removes it from its internal representation of flows.

4.4 Path Selection

Optimal path selection at the end host is becoming an increasingly interesting problem domain as heterogeneity is more pervasive and users and hosts want to maximise the use of their connectivity. Path selection in the core of the Internet has been ubiquitous for a long period of time with technologies such as Equal Cost Multipath Routing (ECMP) [160] and Border Gateway Protocol (BGP) route

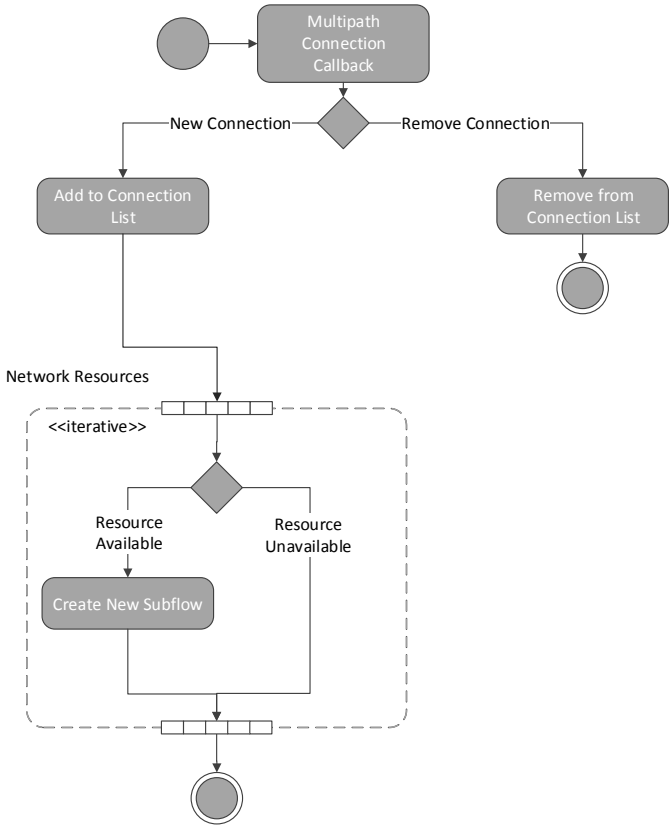


Figure 4.20: Application flow when a new MPTCP connection is established.

selection [141] dictating how to balance traffic. These types of policies however are still not typically present at the edge, more recent work [91] is beginning to address these problems. Historically, path selection solutions have been based on selecting the optimal link for the host to allocate all its traffic to, under an "always best connected" model. With the diversity of connectivity available in a MAP enabled mobile network, selecting the most appropriate network resource for individual flows or applications becomes more promising. Such a fine grained approach can improve the QoS for specific, high priority applications, and potentially reduces cost by preventing over-allocation of delay tolerant, bulk traffic to less expensive network resources.

The core goal of the path selection algorithm is to choose the most appropriate network resource that is able to meet the QoS requirements. This may not necessarily be the absolute best network resource, but that which is best suited for the application. Matching applications to an appropriate network resource can help to distribute traffic across the set of available paths, as opposed to choosing the same path for every application. Due to the multi-dimensional nature of appropriate network metrics and QoS criteria, it is non-trivial to determine the optimal network resource for each individual application. Moreover, the relationships between the specified QoS criteria are typically complex, correlating with one another, limiting the efficacy of simple utility functions and weighted averages as have been proposed in other selection domains for some time [90]. To account for the correlation between measurements, and produce a better estimate of the overall QoS value of each path, we propose the use of Principal Component Analysis (PCA). This approach has been widely used for selecting optimal services, including web service selection, as presented in [90] and [133]. Principal component analysis uses an orthogonal transformation to extract a set of linearly uncorrelated variables (principal components) from the set of original network resource measurements. Removing the correlation between variables allows more accurate decisions to be made, as each attribute becomes independent. As we are not simply looking for the globally optimal resource, we also look to balance across the available links, to do this, we take into account both the QoS of the available paths and the desired QoS of the application. This prevents applications that have minimal QoS requirements from selecting higher quality paths, that are best reserved for applications with more significant network requirements.

4.4.1 Selection Algorithms

The Simple Additive Weighting (SAW) [1] method is a typical selection approach, which attempts to determine the optimal service from a set of criteria. SAW selection algorithms assume that the QoS criteria that are being evaluated are independent of one another; this assumption is not accurate for network metrics as increases in delay, loss or jitter can decrease throughput, especially in the context of a TCP flow. Additionally, relationships may exist between availability, reliability, performance, and cost; further increasing the complexity required to determine the most appropriate network resource. Furthermore, the SAW approach relies on the user or an administrator deciding on a set of weights to determine how important specific criteria are to the application. The selection of weights is of pivotal importance to the success of the algorithm; however, due to the relationship between the proposed criteria, accurately determining the weights is a hard problem. Subsequently, weighting related criteria may cause a users preferences to be counted in a computation multiple times, as the potential correlation is not accounted for. A PCA based approach can eliminate these issues, by extracting the principal components, converting the set of correlated metrics, into a set of uncorrelated linear metrics. In Chapter 3, alternative network and path selection approaches were presented. In the context of the ABC model, the selection approaches typically focused on the allocation of a single network resource observing a global parameter set to describe the best possible QoS. Alternatively, path selection schemes for SCTP primarily use bandwidth as the deciding metric, concentrating on the necessary requirements to extend SCTP itself; additionally this does not account for the range of transport protocols that users may require. Finally the path selection approach proposed in [91], may not meet the level of granularity we strive for; users are split into categories based on how much they are willing to pay for data, however in a pure and flexible path selection approach, the user may be happier to pay varying amounts for a guaranteed QoS for different services, which should be representable in any algorithm.

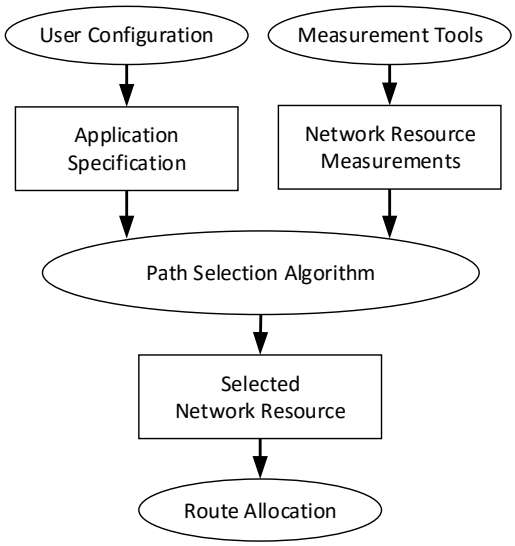


Figure 4.21: Overview of the path selection process.

4.4.2 Selection Interface

As with many of the components we have discussed with the User Policy Framework, the path selection algorithm that is used may heavily depend on the context it is being used. While we propose a PCA based approach in this thesis, to engage with future research and to ease the development of future policy based heterogeneous access models, the propose framework should support pluggable and adaptable path selection. Therefore, the path selection approach should effectively become a black box to the User Policy Framework, in which a set of inputs are provided relating to the available network, and a single selection is provided as the output. An overview of the design for this path selection process is presented in Fig. 4.21. This figure shows the inputs of the system, which is the set of network resources to be selected from, along with their associated measurements and metrics, and the application specification as previously discussed which describes what the optimal set of criteria for the application to perform.

4.4.3 Selection Algorithm

The aim of the PCA algorithm is to transform the original set of quality metrics $q = (q_1, q_2, \ldots, q_j)$ into a lower dimensional co-ordinate space, $Y = (Y_1, Y_2, \ldots, Y_m)$,

Link	BW (Mbps)	RTT(Ms)	Loss (%)	Jitter (Ms)
L1	7.15	141.62	0.20	91.83
L2	10.72	65.84	0.03	14.94
L3	15.63	124.64	0.05	50.51
L4	2.72	166.22	0.26	55.52
L5	10.72	46.97	0.20	22.80
L6	12.01	77.31	0.13	21.43
L7	6.49	115.32	0.14	31.06
L8	1.19	328.02	0.27	151.28
L9	18.32	39.92	0.25	16.82
L10	1.85	362.37	0.10	135.28

Table 4.2: Emulated link characteristics for PCA Path Selection Algorithm example.

	BW (Mbps)	RTT(Ms)	Loss (%)	Jitter (Ms)
1. Voice	0.128	70.000	0.010	30.000
2. Video	8.000	50.000	0.010	30.000
3. Sensor	0.010	300.000	0.00	300.000
4. Unspec	-	-	-	-

Table 4.3: Application specifications and the links chosen from Table 4.2 for communication, according to the PCA algorithm.

while attempting to retain as much information as possible regarding the original quality metrics. During our explanation of the algorithm we provide an example case based on the link metrics in Table 4.2 drawn from work presented in [24] and the applications specified in Table 4.3. In Table 4.3, we present the application QoS specifications that we use as the input for the PCA algorithm. The appropriate criteria for voice and video have been drawn from [23], while the sensor application requirements have been drafted as a low bandwidth low priority service; for example, real time tracking of GPS coordinates. The unspecified application may not have known requirements and is simply looking for the best possible path to be used, therefore the set of application requirements, θ is $\varnothing$. We focus on the performance metrics associated with a specific connection, this could easily be augmented with cost based parameters, such as the price per megabyte in terms of financial cost or battery consumption. For the remainder of this section, both links and network resource are used interchangeably.

Firstly, for a given a set of links, $l = (l_1, l_2, \ldots, l_i)$, each of which possess a set of network metrics, $q = (q_1, q_2, \ldots, q_j)$, we establish the quality matrix Q

shown in Eq. 4.1. This matrix must be normalized according to the min-max function in Eq. 4.2 or Eq. 4.3 as proposed in [90]. The function to be used for normalisation is dependent on the type of metric being normalized. To this end, there are two types of metric that must be supported, those which are better the higher the value, such as bandwidth which are required to use Eq. 4.2, and those which are better the lower the value, such as latency, which will use Eq. 4.3. The normalisation process is required to prevent the variable scaling of different metric measurements from distorting the final result; for example, jitter may be measured in tens of milliseconds, while latency is measured in hundreds. At this stage, if the set of application requirements we wish to optimize against $\theta = (\theta_1, \theta_2, \ldots, \theta_j)$ is not empty, then we begin by calculating the difference between each value in the sets, $q_{i,j}$ and θ_j, as shown in both Eq. 4.2 and Eq. 4.3. The process of normalizing the quality matrix, Q, against Eq. 4.2 and Eq. 4.3 produces the normalized matrix Q'.[1]

$$Q = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1j} \\ q_{21} & q_{22} & \cdots & q_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ q_{li} & q_{l2} & \cdots & q_{lj} \end{pmatrix} \tag{4.1}$$

$$q'_{i,j} = \begin{cases} \dfrac{Qmax(j) - \mid q_{i,j} - \theta_j \mid}{Qmax(j) - Qmin(j)}, & \text{if } \theta \neq \varnothing \\ \dfrac{Qmax(j) - q_{i,j}}{Qmax(j) - Qmin(j)}, & \text{otherwise} \end{cases} \tag{4.2}$$

$$q'_{i,j} = \begin{cases} \dfrac{\mid q_{i,j} - \theta_j \mid - Qmin(j)}{Qmax(j) - Qmin(j)}, & \text{if } \theta \neq \varnothing \\ \dfrac{q_{i,j} - Qmin(j)}{Qmax(j) - Qmin(j)}, & \text{otherwise} \end{cases} \tag{4.3}$$

$$Q' = \begin{pmatrix} q'_{11} & q'_{12} & \cdots & q'_{1j} \\ q'_{21} & q'_{22} & \cdots & q'_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ q'_{i1} & q'_{l2} & \cdots & q'_{ij} \end{pmatrix} \tag{4.4}$$

[1] An alternative approach to calculating the difference between the required QoS and the estimated network metrics, would be to weight each of the criteria based on their importance as proposed in [134]. Weighting however is deemed to be out of scope for this work, as this requires an in depth understanding of the correlation between metrics and their overall impact on the QoS received.

	BW	**RTT**	**Loss**	**Jitter**
BW	1.000	0.231	0.212	0.196
RTT	0.231	1.000	0.169	0.936
Loss	0.212	0.169	1.000	0.235
Jitter	0.196	0.936	0.235	1.000

Table 4.4: Correlation coefficient matrix for QoS metrics, normalized against the video application requirements.

$$cov(x, y) = \frac{\sum(x - \bar{x})(y - \bar{y})}{n - 1} \tag{4.5}$$

$$var(x) = \frac{\sum(x - x)^2}{n} \tag{4.6}$$

$$r_{l,n} = \frac{cov(q_i, q_j)}{\sqrt{var(q_i)}\sqrt{var(q_j)}} \tag{4.7}$$

$$R = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{l1} & r_{l2} & \dots & r_{ln} \end{pmatrix} \tag{4.8}$$

Once we have calculated the normalized matrix Q' (Eq. 4.4), the next step is to calculate the correlation coefficient matrix R (Eq. 4.8) according to Eq. 4.7. The correlation coefficient for our example is presented in Table 4.4. Next the eigenvalues of the matrix R are calculated according to the characteristic equation, $det\,(\mathbf{R} - \lambda\mathbf{I}) = 0$, the resulting eigenvalues need to be sorted in descending order such that $\lambda_1 \geq \lambda_2 \geq \lambda_p \geq 0$. For each eigenvalue, λ_i, the corresponding eigenvector, $\vec{e_i}$, can be calculated by $(\mathbf{R} - \lambda_i\mathbf{I}) \cdot \vec{e_i} = 0$ and normalized based on Eq.4.9. For the normalized eigenvectors (referred to as loadings), the larger the loading the more important the metric is, to that principle component. For example, in Table 4.5 on the next page, jitter is the most significant metric for the first principle component.

$$\hat{e}_i = \frac{\vec{e_i}}{|| \vec{e_i} ||} = \frac{\vec{e_i}}{\sqrt{\sum_{j=1}^{k} \vec{e}_{ij}}} \tag{4.9}$$

QoS Metric	PC1	PC2	PC3	PC4
BW	0.299	0.626	0.719	0.038
RTT	0.641	-0.304	0.036	-0.703
Loss	0.287	0.661	-0.691	-0.059
Jitter	0.646	-0.281	-0.061	0.707

Table 4.5: Loadings for each principal component.

Principal Component	Eigenvalue	CR	CCR
1	2.125	0.531	0.531
2	1.023	0.256	0.787
3	0.791	0.198	0.985
4	0.060	0.015	1.000

Table 4.6: Contribution Rates (CR) and Cumulative Contribution Rates (CCR) for each eigenvalue.

Subsequently we compute the contribution rate (CR) and cumulative contribution rate (CCR) for each principal component according to Eq. 4.10 and Eq. 4.11, respectively as shown in Table 4.6. Typically the principal components that account for 85%-95% of the total variance are carried forward as proposed in [56], [90], and[88]; however, there is no objective function to choose the best subset of principle components, an alternative approach is to simply select all eigenvalues greater than one.

$$CR_j = \frac{\lambda_j}{\sum_{j=1}^{M} \lambda_j} \tag{4.10}$$

$$CCR_k = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{M} \lambda_j} \tag{4.11}$$

Finally for each of the selected principal components, we calculate a utility score based on Eq. 4.12. This score describes the overall quality of each network resource for the specified application. To choose the optimal network resource according to the PCA algorithm, the final step simply selects the network resource with the largest utility score, such that the selected resource is the result of $max(Y)$. This score describes the overall quality of each network resource for the

Path	Y1	Y2	Y3	Utility	Rank
L1	0.616	0.118	0.102	0.836	5
L2	0.910	0.159	-0.026	1.043	1
L3	0.734	0.080	-0.090	0.725	6
L4	0.580	-0.014	0.065	0.631	7
L5	0.838	0.032	0.070	0.940	5
L6	0.830	0.066	0.010	0.907	4
L7	0.845	0.106	0.052	0.247	2
L8	0.075	0.056	0.053	0.183	10
L9	0.654	-0.126	-0.015	0.512	8
L10	0.224	0.181	-0.036	0.368	9

Table 4.7: Final utility scores for each principal component and the aggregate score with rankings of the available network resource in relation to the video application.

	Selected Link
1. Voice	L2
2. Video	L2
3. Sensor	L8
4. Unspec	L9

Table 4.8: Links chosen for the applications specified Table 4.3 on page 116 and links from Table 4.2 on page 116 for communication, according to the PCA algorithm.

specified application, the results of which are shown in Table 4.7. To choose the optimal network resource according to the PCA algorithm, the final step simply selects the resource with the largest utility score. In Table 4.3 on page 116 we present the best path from Table 4.2 on page 116 as allocated by the PCA algorithm.

$$Y_m = \sum_{j=1}^{M} CR_j \cdot (e_{j1}q'_1 + e_{j2}q'_2 \ldots + e_{jk}q'_k) \tag{4.12}$$

For each of the Application Specifications presented in Table 4.3 on page 116 the output of the proposed algorithm selects the theoretically optimal link, as shown in Table 4.8

4.5 Summary

In this chapter, we have introduced the Multipath Advertisement Protocol (MAP), which provides a lightweight and scalable design for cooperative resource pooling in a mobile network. The MAP protocol is used to build and maintain a routing overlay on top of the existing network topology, allowing any infrastructure or ad-hoc protocols to be supported. This protocol provides hosts in a mobile network with access to any of the available network resource. Furthermore, we have presented a framework which exploits the additional network resource, by providing fine-grained context based policies, for both access control and path selection. The path selection approach is based on PCA, which extracts principal components from the measurements, helping to remove correlation between variables and therefore choose the most appropriate paths.

CHAPTER 5

Implementation

This chapter discusses the implementation details of the Multipath Advertisement Protocol Daemon (MAPD), User Policy Framework, and path selection algorithm. The MAPD implementation is a realisation of the MAP described in the previous chapter. This chapter begins with a description of the core system architecture, and implementation specific details not covered by the design. MAPD has been implemented for the Linux Operating System, and has been trialed on a number of flavours including Debian and OpenWRT. The implementation of MAPD has been written entirely in C. Following on from the MAPD implementation, we introduce the User Policy Framework, with a focus on providing a flexible and extensible context aware interface for managing network resource. As with MAPD, the core of the User Policy Framework is written in C with minimal use of external libraries. Finally we present the realisation of the PCA based path selection algorithm. The path selection algorithm has been implemented in Python, due to the availability and simplicity of linear algebra libraries. The Python path selection implementation uses the appropriate application bindings to interface with the C-based User Policy Framework.

5.1 Development Environment

During the development, testing and evaluation of the work, two distinct approaches were used simultaneously. This includes both the use of a real-world testbed and a simulation suite, discussed in more detail below.

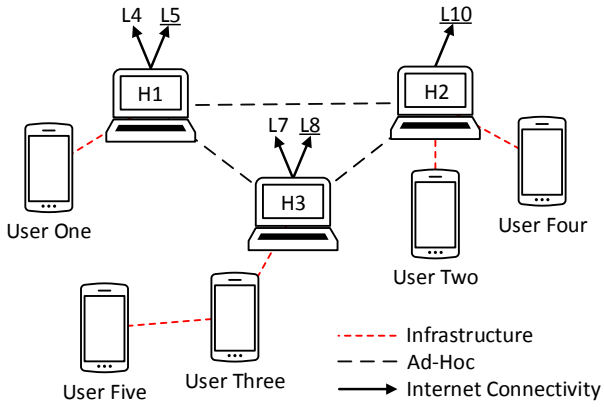


Figure 5.1: Real world experimental development environment.

5.1.1 Real World

To develop and test the proposed system, a real-world testbed has been established. This consists of a set of heterogeneous hosts, ranging from Raspberry Pi's, Wireless routers and Desktop PC's. Each of the hosts in the testbed runs a variant of the Linux operating system, including Raspbian, Debian, Ubuntu and OpenWRT. Each of the hosts in the network possesses one or more WiFi interfaces set in either Infrastructure or Ad-Hoc mode, allowing them to interconnect in complex configurations and furthermore interface with real networks. The typical topology and network configuration used during testing and experimentation is presented in Fig. 5.1. The additional infrastructure access points create a multi-hop environment, requiring MAP to span multiple different networks with different addressing schemes. H1, H2 and H3 run a DHCP server to provide the users with appropriate IP addresses, while they are statically configured with addresses internally within the Ad-Hoc network. Furthermore each of the users can be extended with additional local or Internet connectivity increasing the complexity of the topology. The Internet connectivity is emulated by shaping according to the list of link metrics in Table 4.2 on page 116.

5.1.2 Simulation

To simulate different network topologies with the MAP implementation we use NS3 [118]. NS3 has been extended with the Direct Code Execution (DCE) mod-

ule. The goal of DCE is to provide the user with the ability to easily run existing Linux implementations of user space and kernel space network protocols within the NS3 simulator. Furthermore, the DCE module allows nodes within the NS3 environment to run a real Linux network stack. By using DCE for the simulation portion of the work, it is possible to test and evaluate significantly more complex topologies using the same MAPD implementation. Additionally, by compiling and using our own Linux kernel, we can incorporate MPTCP seamlessly into the NS3 environment.

5.2 Multipath Advertisement Protocol

The MAP Daemon (MAPD) implements the features described in Chapter 4. In this section, we present the realisation of the proposed protocol in a Linux environment, discussing the implementation decisions that were made. For the remainder of this section, both dissemination and advertisement are used when discussing network resources being announced onto a subnet. To further clarify the difference between these two terms, dissemination is the process in which the advertisements traverse multiple hops, spanning the entire network; while advertisement is determined to be the local broadcast or multicast, on a single subnet.

5.2.1 Routing Overlay

In this section we will demonstrate how the routing overlay is built and what this looks like in terms of Linux routing tables and forwarding decisions across multiple hosts. As described in Chapter 4, in order to enable the host to use any of the indirect network resource with resource pooling technologies such as MPTCP, it is necessary to give the host an IP address that it can use to access the remote network resource. To initiate this process, the host in possession of an Internet connection must first create a subnet that allows any other host to route traffic to it. At this stage the host is tasked with allocating a subnet in the 172.16.0.0/12 range. To minimise the chance of collisions when multiple MAP-enabled networks merge, the subnet is chosen randomly, as opposed to incrementally allocating from a base value.

The next step in the process is to ensure all traffic for a given IP address is sent

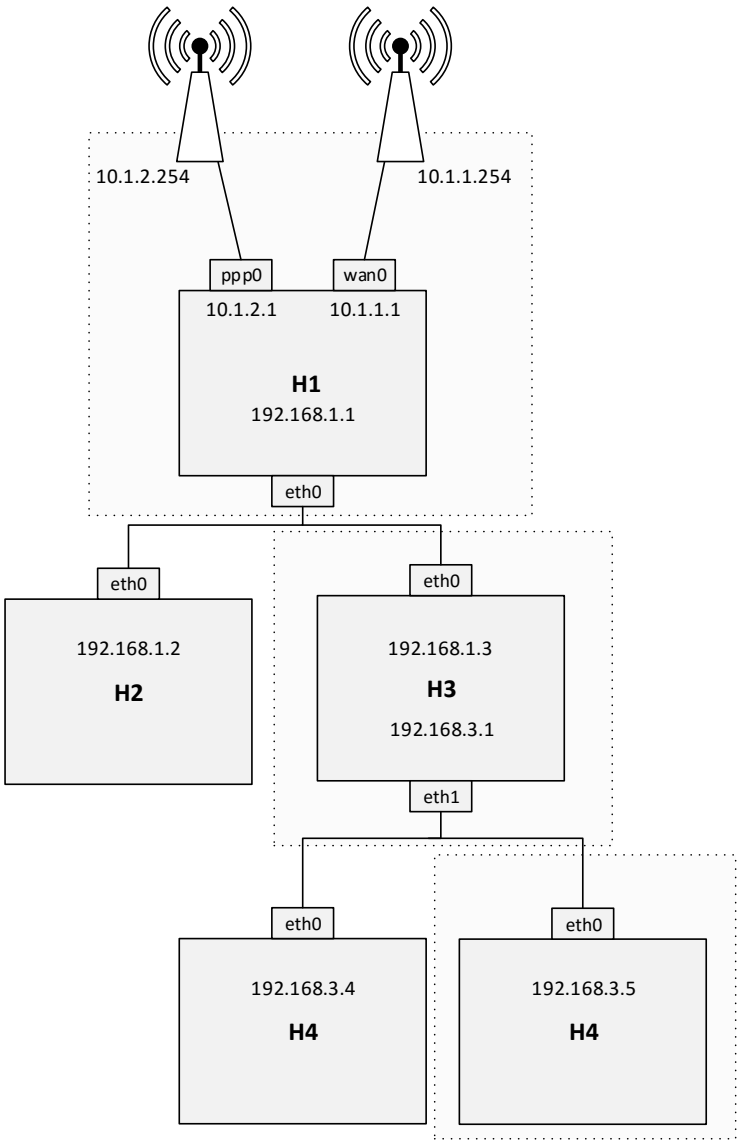


Figure 5.2: Initial addressing routing installed in a simple mobile network.

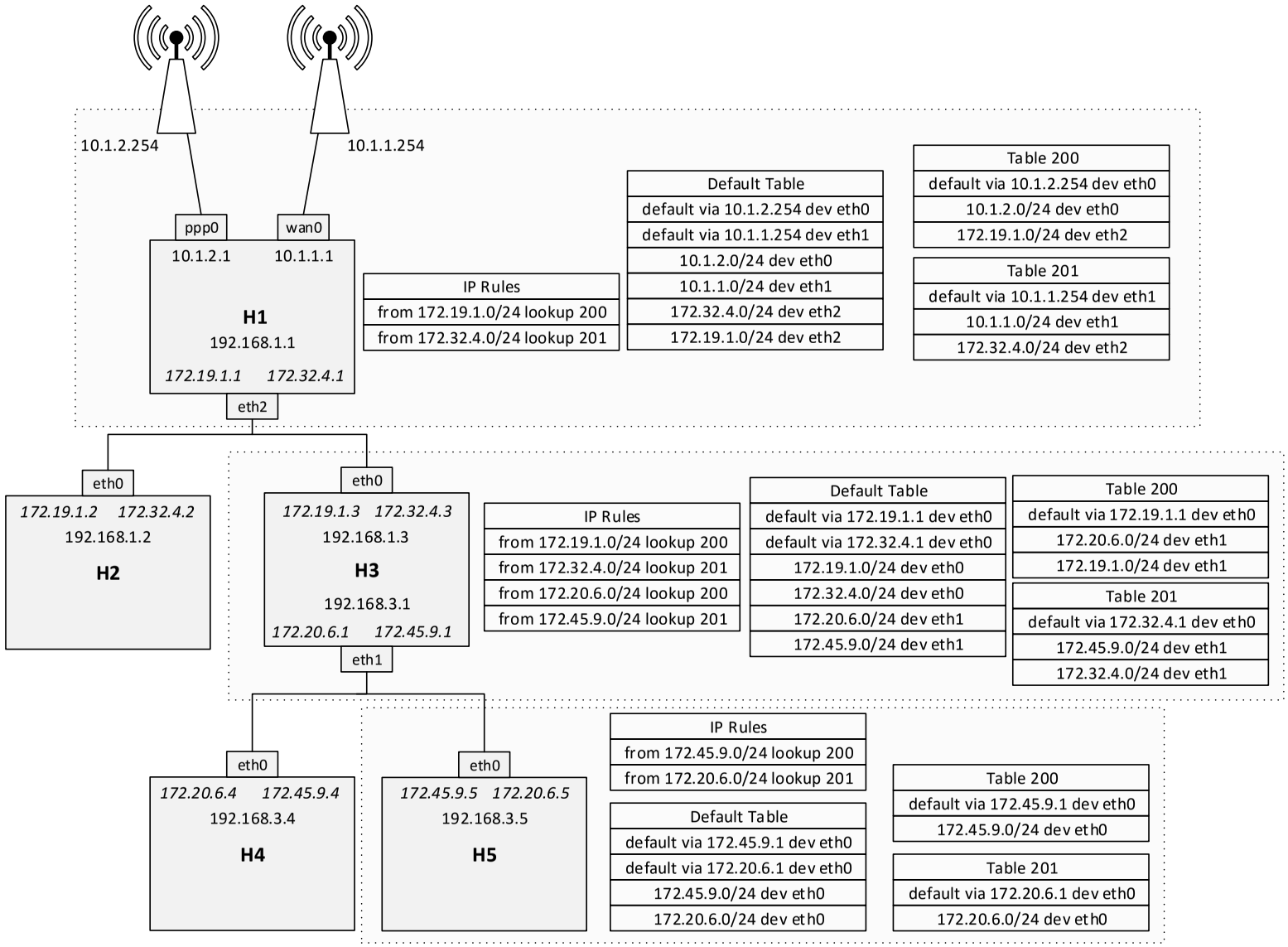


Figure 5.3: Routing tables for Fig. 5.2 after MAP has converged, allowing all network resource to be accessed.

over the corresponding Internet connection, this is achieved through the use of IP rules and IP routing tables, as provided by the Linux Kernel. As of Linux 2.2, up to 255 different routing tables have been supported. These routing tables can be selected based on predefined routing or forwarding rules. This allows different types of traffic to be routed in different ways depending on the specified rules. To ensure this process works across multiple hosts, rules and tables need to be setup to push traffic from one subnet to another, as well as to the final Internet connection. To this end, the host is only concerned with the next hop required to reach an indirect network resource and not the path it takes.

The process of building the overlay is shown in Fig. 5.2 on page 125 and Fig. 5.3 on the previous page. In Fig. 5.2 on page 125, the network has formed allocating IP addresses in the 192.168.0.0/24 range to allow the hosts to communicate directly, this process could occur through either DHCP or static configuration as described in Chapter 4. The only network interfaces offering Internet connectivity reside at the root host, this means that in this example all routing information is disseminated downwards, therefore no routes are advertised back up.

In Fig. 5.3 on the previous page, the MAP network has converged, so every host in the network has a view of the available Internet connectivity in the network, as well as the necessary routing information to reach the gateway. To support this, it can be seen that H1, has randomly allocated the additional subnets 172.19.1.0/24 and 172.32.4.0/24. The additional subnets and the appropriate connectivity information is then passed on to hosts, H2 and H3. H2 does not have any dissemination enabled interfaces, once the update from H1 has been processed, H2 stops. H3 has an additional dissemination interface so it must re-broadcast the MAP update. Before H3 performs the broadcast it must first configure itself according to the update it received. This process includes deserialising the update packet, and adding each of the subnets that were provided by H1 with the addresses and rules for the appropriate routes to access H1's Internet connectivity, as shown on the right hand side of Fig. 5.3 on the previous page. The default routing table has all the appropriate routing information included, for each subnet, while table 200 and table 201 only contain the necessary routes for the subnet they refer to. As H3 has a dissemination interface, the routing and rules become slightly more complex as H3 also requires hosts connecting via eth1 (H4 and H5) to be able to access wan0 and ppp0 at H1. To do this, two more subnets are randomly allocated by H3 for eth1. As H3 has no direct Internet

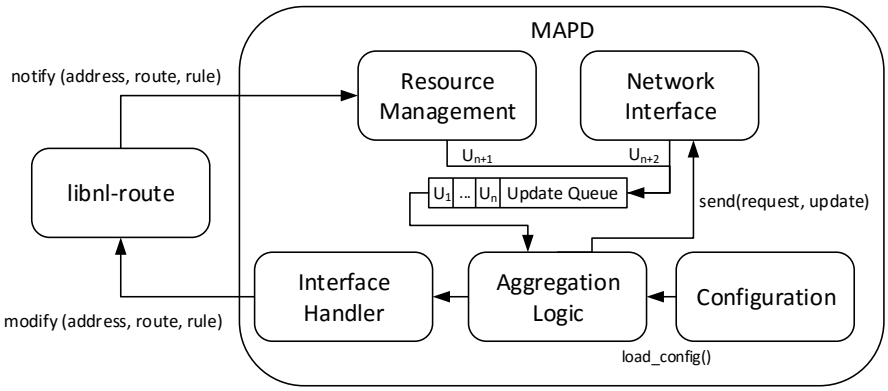


Figure 5.4: The basic architecture for the MAPD implementation.

connectivity, and is relying on H1, the routing tables for these connections are already available and simply need to be augmented with the additional subnets, as well as installing the corresponding rules to forward between the two subnets. A new update packet is then created by H3 and broadcast onto eth1, allowing H4 and H5 to configure themselves in the same way as H3 in stage one.

Each host implements Network Address Translation to avoid the complexity of introducing an additional routing protocol over the top of the existing network infrastructure, this however is not out of the scope of a potential implementation and could reduce the performance overhead required to forward a packet at each hop in the MAP network. Furthermore, the MAP based approach could be incorporated into a routing protocol such as OLSR, however this limits the deployability of the protocol in real world environments.

5.2.2 Architecture

The core of MAPD is shown in Fig. 5.4, this figure shows the main components of the implementation and how they are interconnected. The source code structure that corrseponds to this architecture diagram is presented in Table 5.1 on the next page.

5.2.2.1 Host Configuration

Considering a multihomed host, the user may only want a single interface to be configured for multipath or load balancing, alternatively, the user may not want

Component	Source (.c & .h)
Aggregation Logic	map
Resource Management	link_monitor
Interface Handler	interface
Network Interface	network
Utility	util list queue config

Table 5.1: MAPD implementation components and the corresponding source code.

to advertise on a particular network interface; for example, if the user connects to a WiFi access point, by disseminating link information upwards to the gateway, other MAP capable hosts connecting to the same access point could also access the users primary network resource, such as their cellular interface. Considering the economic and energy costs for some network interfaces this behaviour may not be optimal. To account for this, the initial stage in the setup of MAPD is configuring the interfaces and their associated properties. The policies described by the MAPD configuration are applied independently and are not directly related to those defined by the User Policy Framework. There are two primary behaviours that are configured before MAPD starts, the first is configuration and the use of an ignore list, this is the list of local network resources that the user does not want to be touched or configured by MAPD, this could include expensive links that the user wants exclusive access to, or if a tunnel is configured, the user may only want to advertise the tunnel interface as opposed to the raw interface. The second key configuration property is the dissemination list, this is the list of interfaces that the user wants to advertise their network connectivity options on to. This could be the Bluetooth or USB network interfaces on their mobile devices to improve the connectivity options of their own Personal Area Network, or the WiFi interfaces if the user wants to share their connectivity with others. The use of the dissemination and ignore lists provides an additional benefit of being able to change the topology of the overlay network. By default if all interfaces are included as acceptable for dissemination a mesh topology will form. Previous research has shown that mesh topologies are not always optimal for mobile networks [110], this is especially true of larger networks. If the MAP network is operated by a single entity, such as a fire service, the dissemination interfaces can be configured such that a tree topology is formed as the route overlay. The configuration component uses libconfig [80] to provide a simple format for defin-

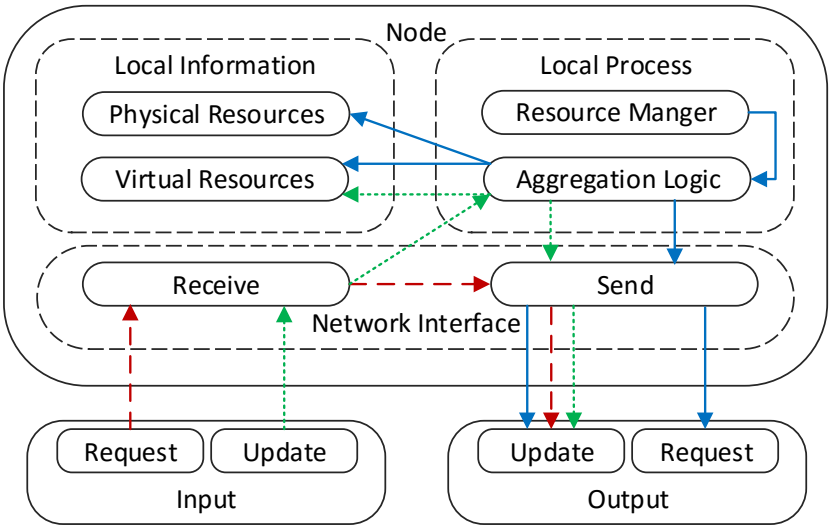


Figure 5.5: The basic flow of data from the network interface and resource monitor in the MAPD implementation.

ing and processing the required data. The interaction between the configuration component and the aggregation logic is shown in Fig. 5.4, the configuration is loaded and imported into the aggregation logic as soon as the daemon starts.

5.2.2.2 Interface Lists

The core data structures for MAPD, consists of two lists, a list for physical interfaces such as WiFi (wlan0), Cellular (ppp0) or Ethernet (eth0), and a list for the virtual interfaces representing the indirect network resources, from hosts offering Internet access. Fig. 5.5 presents a diagram of how these structures interact with the components within the MAPD software, furthermore Listing. 5.1 on the next page presents a minimal representation of the associated data structures which can be found in ***src/interface.h***. The physical list contains direct network resource, and is therefore filled with data in the form of the ***DirectResource*** structure, while the virtual list represents indirect network resource, it is populated with data in the form of the ***IndirectResource*** structure, as shown in Listing 5.1 on the next page. It is important to note that a network tunnel (either layer two or three), is still considered to be a physical interface as it is directly associated with the host.

```
struct NetworkResource {
  int type;
  uint8_t index;
  char name[IFNAMSIZ];
  uint32_t address;
  uint32_t netmask;
  uint32_t gateway;
  uint32_t broadcast;
  uint32_t metric;
  uint8_t table;
};

struct DirectResource {
  struct NetworkResource super;
  uint8_t advertisement;
  List* virtual_list;
};

struct IndirectResource {
  struct NetworkResource super;
  uint32_t sender;
  uint8_t depth;
  struct DirectResource* attach;
  struct DirectResource* out;
  struct IndirectResource* linked;
};
```

Listing 5.1: Direct and Indirect network resource data structures that underpin MAPDs topology representation.

To ease the implementation process we leverage polymorphism and inheritance, therefore both ***DirectResource*** and ***IndirectResource*** data structures inherit from the ***NetworkResource*** parent structure. This allows the different types of network resource to be handled in the same way when possible, and be processed differently when necessary. The physical list of ***DirectResource*** is generated at startup, while the virtual list is then subsequently built around it. Each physical interface has an internal list pointing to the virtual interfaces that are associated with it. For example in the context of H1 in Fig. 5.3 on page 126, ***eth2*** is the ***DirectResource***, while ***172.19.1.1*** and ***172.32.4.1*** represents the ***IndirectResource*** as it provides the subnet for other hosts to reach ***ppp0*** and ***wan0***. Therefore ***eth2***'s virtual list, would contain both ***172.19.1.1*** and ***172.32.4.1***. In the context of the ***IndirectResource*** for ***172.19.1.1***, the ***sender*** field would be empty, as the network resource is local. The ***depth*** field would be 0, as there are no hops inbetween ***eth2*** and ***ppp0***. The ***attach*** variable would point to eth2's ***DirectResource***, while the ***out*** variable will point to ***ppp0***'s ***DirectResource***. The ***linked*** field is not used, as H1 does not have access to any indirect network resource. In the case of H3, the ***linked*** field for the indirect resource ***172.45.9.1*** would point to the structure for ***172.32.4.3***. As before, the attach field would still point to ***eth1***, while the ***out*** field points to ***eth0***, the sender field would be set to ***192.16.1.1*** and the ***depth*** would be incremented to one.

5.2.2.3 Resource Management

The resource management component actively monitors the state of the local network interfaces on a host. The monitoring process is achieved through the use of libnl's route module, which provides a simple interface to the routing subsytem in the Linux kernel. The resource management component is responsible for initially pre-loading the physical interface list with the local network interfaces, if a new Internet connection is added or an old connection is removed the resource manager captures the change and adds it to a queue to be processed. To obtain these messages from the Linux routing subsystem, the resource manager is launched as a thread, registering itself for updates from the libnl-route library. While the resource manager is running, it actively monitors all of the available network resource, at each layer, from link layer MAC address, to network layer

IP addresses and finally network layer routing, filtering for default routes. The resource manager focuses on being provided with a new default route to identify Internet connectivity. If the link address, network address or Internet connection update was added by MAPD itself, as opposed to an external change in system state, the update is ignored when it is passed on to the aggregation logic, which compares all local updates to the current set of known network information.

5.2.2.4 Network Interface

The networking component is responsible for sending and transmitting the connectivity information in the form of MAP update packets. For each interface that the user is interested in receiving updates on, a request packet is sent out, subsequent request packets are transmitted periodically until a response is received. Once an update packet is received, a flag is set indicating that MAP capable hosts exist on the link. The interaction with the network is presented in Fig. 5.5 on page 130, showing how requests and update packets are processed and subsequently move through the system. When a request is received, an update packet is sent back to the requesting host, containing the known network resources. When an update is received, the data structures containing known information are first updated, then the update is re-broadcast onto the other available advertisement interfaces.

5.2.2.5 Aggregation Logic

The aggregation logic essentially provides the glue between the resource management and networking components. The core of the aggregation logic is built around a blocking queue, whenever the resource management or network interface components have an update regarding changes to the available network resource, this is added to an update queue with the associated information, including the type of update and an instance of the update object, for example the default gateway that was added or a copy of a deserialized update packet. The aggregation logic simply processes the update and passes the information to the interface handler, which subsequently makes the appropriate requests to libnl-route, which in turn adds or removes the addresses and routes as is specified by the update.

5.2.3 Resource Pooling

In Chapter 2 a wide range of resource pooling and bandwidth aggregation techniques were introduced. To this end, by exposing a set of IP addresses any of the techniques should theoretically be applicable. In this section we discuss two approaches that have been implemented and tested within MAPD, including load balancing and MPTCP.

5.2.3.1 Load Balancing

As MAPD is running on top of the Linux network stack, applications will not take advantage of the additional links that are exposed, regardless of whether they are direct or indirect. To support load balancing, it is not enough that the default routes are added to the default routing table, as well as their own routing table for the subnet. Load balancing in Linux 3.14 must be explicitly defined by the user to take advantage of additional routes that are available. If MAPD is configured to support load balancing at compile-time, this process is automated. When a default route is added to the host, whether it is a direct or indirect network resource, MAPD identifies this and creates a load balanced default route with a zero priority so it will always be used first. If the load balanced default route already exists, MAPD adds the new default route to the load balanced route, and for route deletions the reverse happens deleting the old default route. This process is limited by a race condition, when adding or deleting a route from the load balanced default route, it is necessary to first delete it, amend the available next hops and then re-add the route. During this time, any TCP flows that are created will not be able to use the load balancer and will fall through to the default route with the next highest priority. As the load balanced default route uses priority zero, all other local routes configured by the user must be set with a lower priority metric, so they are only used by the host when the load balanced route is not present. The limitation of load balancing however, is that applications are still not made aware of the additional routes they can use unless they are explicitly probing the routing tables and network interfaces, this means that the utilisation of all the available network resource is dependent on the creation of enough TCP flows to saturate the available links. Creating a single TCP flow can at best only saturate a single link, assuming it is network limited and not rate limited by the application. This limitation is solved

through the realisation of the resource pooling principle, MPTCP.

5.2.3.2 Multipath-TCP

As Multipath-TCP is effectively implemented as a sublayer between the applications socket and TCP, relying on the available IP addresses to establish new subflows; integration with MAPD is therefore implicit, given the addresses, routing, and rules have been setup appropriately. This is a much simpler approach than was necessary for load balancing to be implemented, with the added benefit that MPTCP will take advantage of all the advertised network resource, no matter how many connections are created by the application. Furthermore, the applications don't need to be modified to see this benefit.

In Chapter 4, we described a limitation of MPTCP regarding the maximimum number of subflows that can be associated with a single pair of addresses and the whole connection, which is set to 8 and 31 respectively when using the default "full mesh" path manager. To address these problems, we have created a patch for the MPTCP kernel module. This patch increases the core number of subflows that the MPTCP kernel can support from 31 to 63. In addition to this core increase, we have also modified the full mesh path manager, increasing the maximum number of subflows that can be attached to a single pair of IP addresses, from 8 to 63. This patch will allow an end-host within a mobile network, communicating with a single homed server, to create and use up to 63 subflows with any of the proposed path managers, which could theoretically provide a significant increase in performance.

The presented modifications to the MPTCP implementation are simple, based on the 0.89 MPTCP release. The most significant change required is to the mptcp.h include file, which contains the function ***mptcp_for_each_bit_set***, which is used to iterate over the set bits in a bitmask, identifying the subflows which are associated with a connection. This function relies on ffs (find first set), which returns the index of the first bit set in the provided bitmask. This function will not work for a bitmask greater than 32 bits (on a 32 bit architecture). Therefore we split the ffs operation into two components, first the bitmask is compared to the maximum value for a 32 bit unsigned integer, if it is less than, the ffs function can safely be used on the lower half, returning the correct answer. If the lower half of the bitmask is equal to the maximum possible value, we run ffs on

the upper half, adding 32 (for the 32 set bits in the lower half) to retrieve the correct answer. This approach allows MPTCP to use a 64 bit bitmask, without introducing any significant overhead when searching path indexes. The next step is to increase the amount of memory allocated to the path manager, so there is space available to store references to the additional subflows. Finally, in the Full Mesh Path Manager, we increase the size of the bitmasks used from 8 bits to 64. As the first bit is used to identify the metasocket for the MPTCP connection, 63 bits are left to identify subflows.

5.2.4 Implementation Decisions

In this section, we will focus on additional implementation specific behaviour that is included in the MAPD software [175]. These behaviours are a matter of preference as opposed to mandated functionality.

5.2.4.1 External Link Identifiers

As discussed in the Chapter 4, external identifiers are needed for each Internet connection, to prevent the formation of loops in the overlay topology. When implementing the protocol with IPv4 there are two options for choosing appropriate identifiers, either the external IPv4 address or the MAC address, both of which should be globally unique. There are however advantages and disadvantages to each, with different use cases. Furthermore, all MAP enabled hosts in the network should use the same type of external identifier, to ensure conflicts are discovered.

IP Address

When using the external IPv4 address for the interface, this IP may not be made available to the host, typically WiFi access points and cellular networks offer addresses in the private ranges, such as 172.16.0.0/16 or 192.168.0.0/24. In order to obtain the external IP address a remote service is required, which can return the IP that was used to connect, bypassing any NAT boxes that reside in between the requesting host and the server.

MAC Address

As the MAC address is available locally, the implementation is much simpler and the address can be retrieved through the same interface as the local IP addresses.

When using the MAC address however, this does not guarantee that the last mile link is not the same. For example, two hosts at the edge of the MAP network could connect to the same WiFi access point, which will generate a conflicting external IP address but distinct MAC addresses. While this still helps remove loops from the MAP overlay topology, it does not stop routes being advertised with the same first-hop as a bottleneck, which may not provide any improvements to bandwidth and add unnecessary congestion to an internal path in the mobile network.

5.2.4.2 Heartbeats and Link Timeouts

In MAP, there is no strict rule defining hosts or gateways as may typically be found in an infrastructure or Ad-Hoc network. Looking back to the use-cases defined in Chapter 1, a number of smart phones may be connected to a Wireless Access point on public transport. Each of these smart phones has access to a cellular interface in addition to Internet connections made available by the Wireless Access Point. In this scenario, if the user has made their cellular interface available using MAP, all of the MAP capable hosts in the network can also use this link, in addition to their own links and the links provided by the Wireless Access Point. As this network topology is blurring the line between the definition of a host and a gateway in an infrastructure based network, ensuring a link is still available becomes more challenging. When a client disconnects from a Wireless Access Point, the client side behaviour is reasonably well defined, typically when using software such as WPA Supplicant [107] in a Linux environment, this process will trigger the interface to remove its IP address, which in turn will register an event with MAPD. On the access point side of the connection, the removal of the client is not exposed in the same way. In order to detect these events, and disseminate opaque link losses, MAPD periodically sends out heartbeats to inform the network that the available links are still active and available to be used.

The heartbeat packets are not disseminated any further than the directly connected subnets, to ensure that they do not end up flooding the network, as each host would be replicating that nothing has changed for each hop. If an update packet is received, this means that the host is still active and the heartbeat timer is reset. The heartbeat timer is implemented as a thread that runs when the MAPD process is first started, when the heartbeat time is reached

a flag is set, notifying the networking thread that an update must be sent out on each of the dissemination links. To supplement the heartbeat packets, the host also needs to react appropriately if they are not received. Handling the loss of a host, essentially means that the MAP needs to remove all associated links. Each indirect network resource that is made available to a host is given a link timeout value, the link timeout thread then decrements this value periodically and on reaching zero, the link is removed and the change is disseminated to all other interested hosts. As the MAP is only interested in hosts that have direct or indirect access to an Internet connection, hosts without any better information will not send heartbeats, reducing the impact on the network. This lack of better information is enforced by the fact that network updates are not disseminated back on to the originating link.

As the MAP is UDP based, if a single heartbeat is not received, this may not represent the loss of a host, and therefore the link timeout value should be set to a multiple of the heartbeat time. Additionally the use of heartbeats also proves to be useful if the internal network links have a high probability of loss, as the update packets may be lost, allowing the heartbeat to amend for the loss of an update. The heartbeat and link timeout values are configured by the user, the optimal values for this process are highly dependent on the context of the network, if there is a significant amount of churn in the network, such that hosts are disconnecting and reconnecting regularly, then a high frequency of heartbeats and a low timeout value would be better to ensure the information is propagated as quickly as possible, alternatively if the network does not change regularly, then a lower frequency of heartbeats and a longer link timeout value will be better as to reduce the overhead on the network. The decision to use heartbeats in the implementation was to create the most flexible and portable solution, alternative approaches could have included hooking into the management of the network interface, or having the hosts indicate that they are likely to leave the network due to a drop in signal strength or through the user disabling an interface. Providing hooks into the network interface would require significant implementation across the technologies that may be used, such as WiFi, Bluetooth or Ethernet. With some technologies such as WiFi Direct, the disconnection presents as an explicit event, therefore if an implementation was link layer dependent, heartbeats could potentially be removed.

5.2.4.3 Scalability

As the size of the networks that run MAPD grow it may not always be desirable to advertise every Internet connection to every host. For example, if a host is multiple hops away, or if one of the internal paths is of particularly poor quality the benefit of using the indirect network resource may be negligible and use up internal and external resource that a different host could make better use of. Adding hard boundaries to a MAP mobile network would be a coarse grained solution that could prevent multiple hosts from accessing network resource that is pivotal to their communication needs. To prevent this, MAPD provides soft, logical boundaries to network resource advertisement. This means while MAP will have complete coverage of the entire network, not all links will be pushed down if a host deems it not to be necessary. Furthermore, logically segmenting the network will help to reduce traffic internally, for example hosts at opposite ends of the mobile network will not use up all the bandwidth trying to reach an Internet connection located at the opposite host. In the MAPD implementation we have used the hop count to represent the point at which a link stops being advertised, however this logical boundary is considered to be a pluggable metric. Alternative metrics could include looking at the total cost of transmission between two hosts in the network, this could be the bandwidth, signal strength or alternative metrics such as power cost. Alternatively the metric used could be modified based on the external connectivity metrics. A number of alternative metrics and measurements that may be appropriate were presented in Chapter 3.

5.2.4.4 Supporting Multipath Unaware Hosts

A TCP Proxy can be used at each host to transparently add support for Multipath-TCP. This has potential to break the resource pooling benefits when attempting to combine multi-hop networks, as each TCP connection is split at a proxy, the MPTCP connection would also be terminated. This can be circumvented by first checking that the inbound connection to split is not already multipath enabled. An example of the approach required to proxy only standard TCP flows is presented in Listing 5.2 on the next page. This iptables command filters on the TCP options that are set before deciding to redirect a packet to the implemented TCP proxy, via the TPROXY chain.

```
1 MPTCP_KIND=30
2 iptables -t mangle -A PREROUTING -p tcp --tcp-option \\
3 ! $(MPTCP_KIND) -j TPROXY --tproxy-mark $(TPROXY_MARK) \\
4 --on-port $(TPROXY_PORT)
```

Listing 5.2: Transparent redirect to MPTCP proxy for traditional TCP traffic.

5.2.4.5 API

In Chapter 4 the need for an appropriate Application programming interface (API) was justified which allows any process within the system to understand how packets should be marked or routed in order to implement policy routing or path selection functionality. To provide a lightweight API in keeping with the rest of the protocol implementation, the network resource and the appropriate routing decisions are presented by the file system, similar to that provided by Linux's ProcFS. Given that an application has access to a network resource, a simple lookup is all that is required to obtain the routing decision that must be applied to the traffic class to enforce the specific route. The appropriate file structure for obtaining the information is located at "/tmp/mapd". From here, the application checks either the "direct" or "indirect" sub-folders, using the IP address as the lookup identifier for the network resources file. The file matching the network resource that is being searched for then contains the appropriate routing table for packets to be sent to. Using this approach an application can use iproute2 or iptables to install a routing rule that specifies a certain percentage or a certain type of traffic should always be routed via that network resource. This approach while simplistic, removes the complex dependency on RPC-like techniques or requiring the use of sockets to read and write the minimal information set that is needed. This file is written as soon as MAPD acquires the routing information and is subsequently destroyed when the route is removed.

5.3 User Policy Framework

In this section we will present in detail the implementation of the User Policy Framework. In comparison to MAPD, the User Policy Frameworks base structure is much more complex (detailed in Table 5.2 on the next page), as it is based around a flexible and extensible modular interface for context monitoring and

Component	Source
Main Logic	main.c
Resource Management	resource_manager.c link_monitor.c link_loader.c
Network Measurements	link_metrics/ path_metrics/*
Context Management	conditions/* context_library.c action.c
Route Allocation	flow_manager/* iptables/*
Path Selection	path_selection/* application_rules.c
Utility	util.c queue.c list.c

Table 5.2: User Policy Framework implementation components and the corresponding source code.

path selection decisions. The purpose of the User Policy Framework is to provide a rich and descriptive approach to defining how the available network resources are used. This includes determining when the available network resources should be used, and additionally provide a mechanism to optimally allocate traffic to the best network resource. In the previous Section, MAPD presented a minimal set of policies for controlling the usage of local network resources. This is limited to specifying which interfaces can and can't be used for advertisements. The User Policy Framework goes beyond the simple nature of an access control list, dynamically allocating and deallocating network resource for use, based on the current context of the network and the device.

5.3.1 Resource Management

The Resource Management component has a number of roles that are similar in terms of functionality to the MAPD resource manager. There are three main tasks that the Resource Manager should carry out. In the absence of MAPD, it should be able to identify and load the network resources with Internet connectivity. Otherwise, if MAPD is running, the resource manager should interact appropriately to understand the state of both direct and indirect resource. Finally, the resource manager is responsible for obtaining current network measurements regarding the quality of the available network resource.

5.3.1.1 Data Representation

The representation of the available network resource within the User Policy Framework is simpler than MAPD, as we are only interested in network resource,

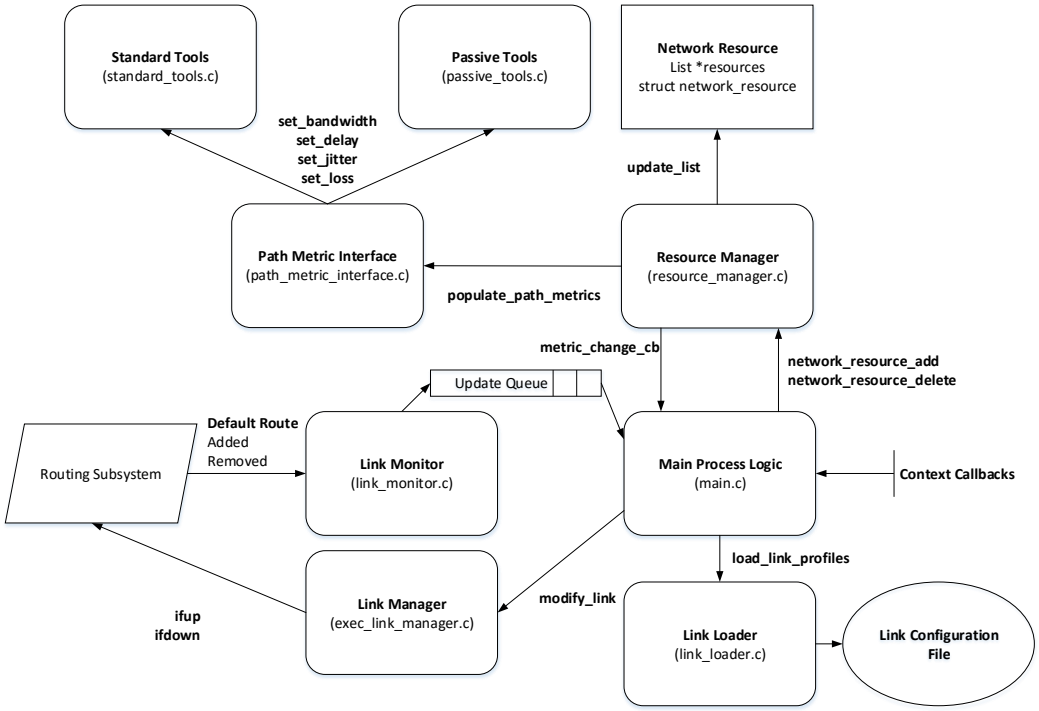


Figure 5.6: Implementation of the User Policy Frameworks Resource Manager component.

the location is of little concern. However the list data structure used is more sophisticated by introducing callbacks when network resources are added or removed, which in turn triggers events within the core User Policy Framework and the Resource Manager. When new resource is added, a thread is created to monitor the quality of the network. The data representation is further augmented with a configuration file, which describes the set of links that are available, and how they may be used by the host. For example, when a network resource is identified, the name is compared to the information in the configuration file, which specifies if the link layer technology used is cellular, WiFi, WiMax or Ethernet etc. Furthermore, the configuration also specifies the multipath capability of each of the links. The default behaviour is for all links to automatically be used by multipath traffic, however using this configuration file, links can be set to act permanently as a backup or to disable multipath behaviour entirely.

5.3.1.2 Link Monitor

The behaviour of the link monitor is a minimal implementation of the resource manager that is provided by MAPD, as the User Policy Framework is only interested in Internet connectivity the additional information regarding associated subnets and the link layer can be safely ignored. As with MAPD the link monitor obtains notifications from the kernels routing subsystem through the libn-route library.

5.3.1.3 Link Manager

The link manager is an additional feature of the Resource Manager that is not present within MAPD. The link manager has the ability to enable or disable a link from being used, which is an important function for the context management component of the User Policy Framework. Unconfiguring a specific link can help to save additional power, in the case of WiFi the network interface will stop scanning for access points reducing the cost associated with having the interface attached.

5.3.1.4 MAP Interaction

The interaction with the MAP API is straightforward, simply requiring the Resource Manager to query the file structure that was previously discussed in order

to discover the routing table that is used to access a specific network resource. Once this process has been carried out, the number for the routing table is stored in the network resources data structure, which is then used by the User Policy Framework when enforcing path selection decisions.

5.3.1.5 Network Measurements

The network measurement portion of the resource manager is implemented as a simple interface, therefore the User Policy Framework has no interest in the implementation of the tools or techniques that are used to obtain the measurements. This approach allows the user or administrator to integrate or implement the most appropriate approach depending on the context the device is used in. This could be a combination of both passive and active metrics, or simply determining the expected measurements based on the type of connectivity. For the actual implementation, we have built interfaces to the iPerf, Abing [114] and Ping utilities to obtain active measurements. Additionally passive bandwidth measurements have been implemented monitoring the number of bytes that have been sent and transmitted on each network interface. The measurements for each network interface are requested periodically, which results in the previously mentioned tools running with the output being parsed before being handed back to the network metric interface. iPerf is heavyweight in terms of the amount of injected traffic, Abing provides a much lighter estimation approach calculating the bandwidth by observing the inter-packet arrival time of a small packet train. Abing however is less accurate than iPerf, especially in the presence of packet loss. As bandwidth estimation approaches improve, the network measurement approach can be transparently upgraded without any core changes to the system.

5.3.2 Context Management

The context management component of the User Policy Framework is responsible for controlling access to network resource based on the provided context. Context is determined by a set of dynamically loaded, pluggable modules. These context modules are paired with a descriptive policy set, which defines the users requirements and needs in terms of how network resource should be accessed.

5.3.2.1 Context Configuration

An example configuration of contexts that the User Policy Framework can monitor and react to is presented in Listing 5.3 on page 147. The configuration is defined in JSON, providing a simple and extensible schema, which could be retrieved via RPC or AJAX from a context repository, or configured by a local user interface. Listing 5.3 shows an example with two policies. The first policy defines a set of conditions, when both conditions are true, the action to disable the WiFi will be carried out. To meet both conditions, the WiFi interface must be connected to the Access Point with SSID "PayPerMB_AP" and have used more than 250 Megabytes of data. Subsequently, the next policy definition causes the cellular interface to turn on, when it is detected that the WiFi interface is no longer in use. This configuration represents a typical model for mobile connectivity, with the additional factor that the handover process is also concerned with the cost of the network interfaces and not only the availability. Furthermore, this model could be extended with any of the proposed contexts such as location or battery capacity.

5.3.2.2 Context Modules

The context modules are built separately to the User Policy Framework and are dynamically loaded at run time by the context manager. To create a context module, the compiled shared library must implement the interface from ***condition.h*** as shown in Listing 5.4 on page 148. The ***init_condition_t*** function is used to initialise the appropriate data within the context module. The minimum behaviour required during initialisation of a context module is to call the ***register_key_cb_t*** function, which passes the list of keys associated with the Context Module back to the User Policy Framework. Each of the keys that are announced specify a unique condition that the Context Module is able to monitor. For example, in Listing 5.3 the key "ssid" refers to the module that checks which Wireless Access point wlan0 is currently connected to. After registering all of the keys, the context configuration specifying the users policies can be loaded. While loading the context configuration, the ***parse_condition_t*** function is used to convert the JSON formatted condition into its software representation. By allocating this functionality to the Context Module instead of the User Policy Framework, the Context Modules can make the condition definition as complex as required

without changing the User Policy Framework. Once the User Policy Framework finishes the initialisation process, ***start_context_lib_t*** can be called for each Context Module, which is used to initialise the condition monitoring process. Finally, the ***condition_cb_t*** function is used to provide the context module with a reference to the User Policy Framework, therefore when a condition is met an event is triggered inside the framework, matching the condition to the associated action.

5.3.2.3 Context Manager

The context manager provides the interface for the main program logic to start and control the contexts that the User Policy Framework is interested in. The first point of call for the Context Manager, is to load all of the Context Modules at runtime that the User Policy Framework has been instructed to use. The Context Manager takes each entry in the list of Context Libraries and sequentially loads them, this process returns a handle for each dynamically loaded library which is stored in the ***context_library*** data structure. Given the handle associated with the Context Module, the ***context_library*** structure is populated by attaching the functions defined in Listing 5.4 on page 148 to the related function pointer. The list of context libraries is then used when loading the context configuration file, as previously described by allowing each Context Module to parse the appropriate conditions based on the provided key value.

5.3.3 Policy Handler

The logic within the policy handler represents the core of the path management and network resource utilisation information. All of the events that the User Policy Framework reacts to ultimately end up being addressed by this component. Therefore it is imperative for the logic in this component to be understandable and easily extensible. In Chapter 4, we presented a number of callback events including:

- Condition Callback
- Resource Change Callback
- Metric Change Callback

```
[
    {
        "policy": {
            "condition":[
                {
                    "link_id": "wlan0",
                    "key_id": "bandwidth_allowance",
                    "value":"250Mb",
                    "comparator":"<"
                },
                {
                    "link_id": "wlan0",
                    "key_id": "ssid",
                    "value":"PayPerMB_AP",
                    "comparator":"=="
                }
            ],
            "action":[
                {
                    "do":"disable",
                    "link_id":"wlan0",
                    "mode":"soft"
                }
            ]
        },
        "policy": {
            "condition":[
                {
                    "link_id": "wlan0",
                    "key_id": "availability",
                    "value":"off",
                    "comparator":"=="
                }
            ],
            "action":[
                {
                    "do":"enable",
                    "link_id":"ppp0",
                    "mode":"hard"
                }
            ]
        }
    }
]
```

Listing 5.3: Example of a policy configuration file specifying the interface wlan0 should stop being used when the bandwidth allowance of 250MB has been exceeded while connected to the AP with SSID “PayPerMB_AP”.

```
typedef int (*init_condition_t)
    (register_key_cb_t reg_cb, void *data);

typedef int (*register_key_cb_t)(char *key, void *data);

typedef struct condition* (*parse_condition_t)
  (void *key, void *value, void *comparator);

typedef void * (*start_context_lib_t)(void *context);

typedef void (*condition_cb_t)
  (struct condition *c, void *data);
```

Listing 5.4: The interface which context modules must implement in order to be loaded and used by the User Policy Framework.

- MPTCP Callback

Each of these callbacks culminates in the policy handler providing an overall network resource management and path selection algorithm. By designing and implementing these events as a set of callbacks, the exhibited behaviour can be segmented into a set of logical components, creating a series of simplified sub algorithms that were presented as activity diagrams in Chapter 4. The overview of the policy handler and the associated components is presented in Fig. 5.7 on the next page; this figure presents the interaction between the callbacks triggered by the User Policy Framework, path selection and route allocation components.

5.3.3.1 Application Configuration

An example configuration for the application context is presented in Listing 5.5 on page 150, as with the context manager this is represented as JSON, and could be obtained from an administrator or optimally an application specification repository. This figure shows the application specification i.e., what the flow should look like that we are interested in matching against, followed by the requirements of that application. At present the requirements configuration is based on the associated network metrics, however cost parameters could easily be integrated. The additional parameter that is introduced aside from network metrics, is the multipath capability. This specifies for each application whether or not multiple paths should be created when possible. This configuration property can therefore

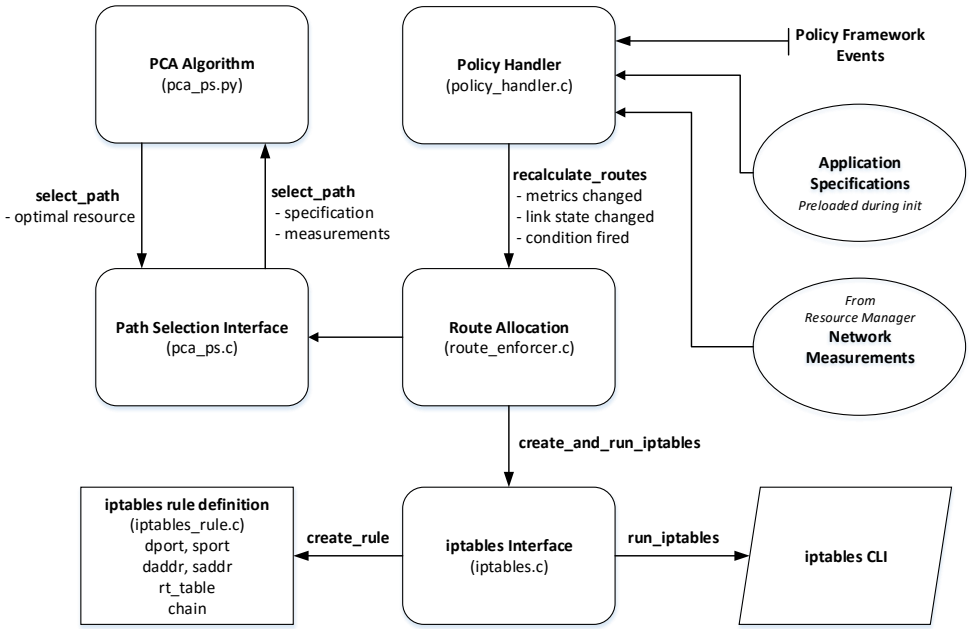


Figure 5.7: Overview of the path selection process within the User Policy Framework.

override the proposal for a socket option controlling this behaviour, which would require all applications to be modified. There are a number of reasons why TCP based applications may choose to avoid using multiple paths simultaneously, for example, delay sensitive applications may prefer to use a single low delay path instead of spreading traffic across multiple paths. Additionally application limited flows that typically gain no benefit from resource pooling may also choose this option. The User Policy Framework presents a unique use case for this feature, as bandwidth estimation tools will only want to use a single link to obtain a realistic representation of the available bandwidth, without having to disable multipath or the additional links. Moreover, an alternative application may be looking to estimate the total available bandwidth using similar tools, therefore, modifying the application may not always be appropriate.

5.3.3.2 Route Enforcement

To allocate and enforce the routes that are chosen by the path selection algorithm via the policy handler, we use iptables to ensure that packets are routed appropriately. The route enforcement process relies on a simple interface to the iptables

```
[
    {
        "spec": {
            "application": {
                    "daddr":"8.8.8.8",
                    "saddr":"",
                    "dport":"80",
                    "sport":"",
                    "proto":"tcp"
                },
            "requirements": {
                    "bandwidth":"1Mb",
                    "loss":"0.01",
                    "jitter":"30.00",
                    "latency":"80.00",
                    "multipath":"enabled"
                }

        }
    }
]
```

Listing 5.5: Example of an application specification configuration file.

```
IPT=$(which iptables)
$IPT -A OUTPUT -t mangle -j CONTEXT
$IPT -A CONTEXT -t mangle -j CONNMARK --restore-mark
$IPT -A CONTEXT -t mangle -m mark ! --mark 0 -j ACCEPT
$IPT -A CONTEXT -t mangle -j CONNMARK --save-mark
```

Listing 5.6: Example of the base iptables rules that are inserted for unspecified applications.

```
$IPT=$(which iptables)
$IPT -I CONTEXT 3 -m mark --mark 0 -s 0.0.0.0 -p TCP \\
 --dport 80 --daddr 8.8.8.8 -m conntrack --ctstate NEW \\
 -t mangle j MARK --set-mark 200
```

Listing 5.7: Example of an iptables rule that allocates a flow to a specific network resource

```
1 $IPT=$(which iptables)
2 $IPT -t nat -A POSTROUTING -m mark --mark 200 \\
3  -j SNAT --to-source 172.16.1.23
```

Listing 5.8: Example of iptables rule to implement source NAT to ensure correct routing.

command line interface. In order for the correct routing to take place using this approach, there are two key parts to the rules we install. Firstly packets must be marked to match an IP rule pointing them to the correct routing table that has been installed by MAPD, if it is running, otherwise the routing table used matches the interfaces index. This first step pushes the packet into the correct routing table, however as the source address will be incorrect at this stage in the routing process, it must be modified to match the associated table. To ensure this process is extensible with any number of rules, without conflicting with the users own rules, we create a new iptables chain, appending a set of predefined rules which are to be used by default flows that don't have an application specification as shown in Listing 5.6 on the previous page. The rules in Listing 5.6 on the previous page specify first that a new chain must be created, then for each packet restore any mark that has been placed on the associated flow, if the mark is 0, this means the application is unspecified, and therefore can be routed using the default behaviour. Listing 5.7 on the previous page presents a specific rule that will cause a flow to be directed over a specific interface. This rule is inserted into the CONTEXT chain after line three (–restore-mark), if the packet matches the configuration loaded from Listing 5.5 on the previous page, all future packets belonging to that flow will be marked in the same way. We use the connection tracking module to ensure we don't break any single path flows that don't support mobility, therefore the mark will only be established for the first packet in a flow. The source NAT rule presented in Listing. 5.8, that circumvents strict firewalls is created when the associated network resource is added to the User Policy Framework, and is subsequently destroyed when the network resource is removed.

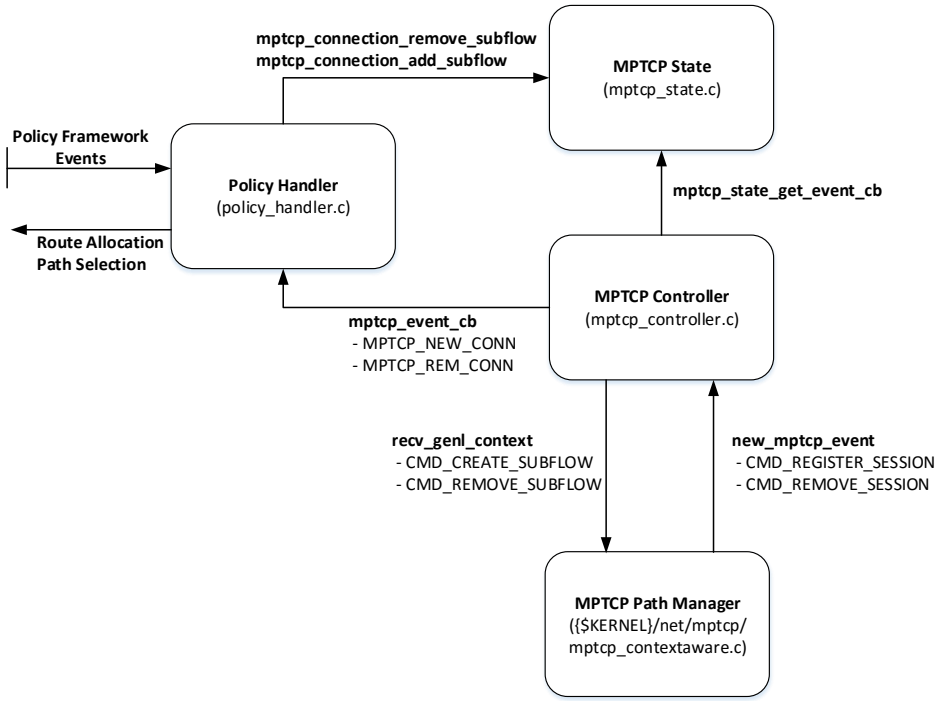


Figure 5.8: Overview of the MPTCP user space controller and kernel space path manager implementation and interactions.

5.3.4 MPTCP Controller

The MPTCP controller is essentially made up of two key components, a kernel space path manager and a user space controller. The user space controller is subsequently called from the User Policy Framework when path management changes need to be made. Additionally the kernel space component is able to notify the user space controller of any changes in state to multipath connections, such as creation or deletion. Our custom implementation of the MPTCP path manager is located in the MPTCP kernel module under ***mptcp_contextaware.c***, while the user space component resides within the User Policy Framework source under ***mptcp_controller.c***. An overview of how the MPTCP controller communicates with the path manager and the User Policy Framework is presented in Fig. 5.8.

5.3.4.1 Communication

To enable communication between the user space MPTCP controller and the kernel space path manager, we use the Generic Netlink protocol. Netlink can

be used to exchange information between the kernel space and user space, as presented by the resource manager in the MAPD implementation, which uses it to obtain notifications to changes in network addresses and routes. The base Netlink implementation is limited in the number of applications that can be used or represented, as each application must define a protocol type in ***include/linux/netlink.h***, which would require the kernel to be re-compiled to introduce changes. Generic Netlink provides a controller which dynamically allocates communication channels to applications and services, using strings as an identifier for registering a new protocol instance. Applications, services and modules can then obtain a reference to the appropriate channel using the pre-determined string identifier. Generic Netlink, therefore removes the limitations imposed by Netlink allowing any application and kernel module to define their own interactions.

When the MPTCP path manager is loaded into the kernel, it registers the "Context Aware" protocol family with the Generic Netlink controller, along with the set of associated functions that may be called from the user space controller. On top of the base path manager functionality, our implementation introduces three additional functions. These functions include:

- ***register_session_with_daemon***
- ***remove_session_from_daemon***
- ***recv_genl_context***

The register session function is called when a new MPTCP session is created, and passes the identifying tuple for the MPTCP connection along with the identifying token for the session up to the MPTCP controller. The remove session function performs the opposite action, passing the identifying token for the connection and alerting the MPTCP controller that the socket is now closed. Finally, the ***recv_genl_context*** function waits on messages from the MPTCP controller. These messages contain the necessary information to create or delete an MPTCP subflow. The required information includes the identifying token for the MPTCP session, the remote hosts IP address and port, and the address of the local interface which the subflow should be associated to. When a modification to the set of subflows in a session is required, a structure containing the appropriate session modifications is added to a list in the path managers data structure, which is then added to a workqueue for processing in the future, when possible.

5.3.4.2 User Space

This MPTCP controller contains the appropriate control information for communicating with the kernel and receiving signals that connections are to be added or removed. There are three important functions that are present within the MPTCP controller, the first is the initialisation function that attaches to the Generic Netlink protocol family that was registered by the kernel space path manager. This initialisation runs as a thread that then waits to receive notifications from the path manager. The second is the ability to receive notifications from the path manager, regarding MPTCP connections to add or remove. Finally, the third is to provide an interface which allows MPTCP connections to be dynamically modified, adding or removing subflows on-the-fly. The MPTCP connection tracking and subflow modification needs to interface with a user space representation of the current set of MPTCP connections and the active subflows that are associated with each connection, so the policy manager can react to the set of proposed callbacks appropriately.

5.3.4.3 Kernel Space Modifications

In the MPTCP kernel module, each session is identified by a unique token. In order for the user space controller to be able to reference and modify a pre-existing MPTCP session, the logical approach is to provide the controller with a copy of the token along with the identifying tuple for the session. Furthermore, the kernel space path manager needs to be able to obtain a reference to a specific session purely using this token. This behaviour is not possible with the current stock implementation of the MPTCP kernel module, we introduce this functionality with a focus on limiting the required changes to the kernel module. In ***mptcp_ctrl.c***, the function (***mptcp_hash_find***) provides an MPTCP socket given the unique identifying token and a reference to the net namespace that is in use. The standard implementation supports an optimisation in which only sockets for the appropriate namespace are checked, however, when the path manager is called from the user space module, this reference is not available. Therefore to obtain the socket, we remove this optimisation, and export the function such that the socket can be retrieved anytime from anywhere in the MPTCP kernel module. It is important to note that changes to performance were negligible, in terms of subflow creation time.

The MPTCP kernel module presents a number of signals that the path manager is able to register, and subsequently react to the events when the signal is fired. These include signals such as a new MPTCP session being created and addresses being added or removed. There isn't however, a signal present for when the MPTCP socket is closed, which is required to alert the MPTCP controller that the session has finished and can be removed from the list. To accommodate this, we add the ***close_sock*** signal to the ***mptcp_pm_ops*** structure in ***mptcp.h***. Then during the call to ***mptcp_close*** in ***mptcp_ctrl.c***, if there is a function registered to the ***close_sock*** signal, it is fired allowing the path manager to react appropriately.

5.4 Path Selection

In this section we will present the implementation of the Principal Component Analysis (PCA) path selection algorithm. The overall goal for this algorithm is to select the best possible resource for a specific application. The measurements for each available network resource are normalized with respect to the requirements of the application, therefore, we add an additional dimension to the selection procedure, preventing the algorithm from choosing the best network resource, and instead choosing the network resource that best matches the requirements of the specified application.

5.4.1 Algorithm Implementation

The PCA approach presented in the design chapter can be summarised as follows:

1. Create quality matrix Q for Network Resources (NR).
2. Normalize Q against normalisation function M.
3. Calculate Correlation Matrix R from Q'.
4. Compute Eigenvalues and Eigenvectors of R.
5. Sort the set of Eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ in descending order.
6. Select n-Eigenvalues to form a loadings matrix L.

7. Calculate the Contribution Rate (CR) and Cumulative Contribution Rate (CCR) of each λ_i.

8. Given CR, Q' and L, calculate utility scores for each NR.

9. Select NR with the highest utility score.

This process is broken down and presented in psuedocode in Alg. 1 and Alg. 2. Alg. 2 demonstrates the general process, while Alg. 1 details the calculation of the contribution rates. This figures illustrates the steps required to choose the optimal network resource for a single application. Therefore the discussed steps must be performed for each application that is presented in the application specification configuration file. The Python path selection algorithm takes two inputs, a list of the available network resources and the associated metrics using the local IP address as an identifier, and the application specification used during normalisation of the network resource. The path selection algorithm then returns the IP address that should be used as the source address for routing the application, which can subsequently be used to determine the appropriate routing table.

Algorithm 1 Algorithm to compute the contribution rates of eigenvalues

Input: E - Sorted list of eigenvalues.
Output: P - List of CR and CCR for each eigenvalue.

```
1: function CALCULATE_CR_CCR(E)
2:     ccr ← 0
3:     cr_arr ← [ ]
4:     ccr_arr ← [ ]
5:     ev_sum ← sum(E)
6:     for i; len(E) do
7:         cr ← E[i]/ev_sum
8:         cr_arr[i] ← cr
9:         ccr ← ccr + cr
10:        ccr_arr[i] ← ccr
11:    return cr_arr, ccr_arr
```

5.4.2 Integration

To implement the path selection algorithm, we use python to take advantage of the flexible and easy to use linear algebra libraries available. Using the standard

Algorithm 2 PCA Path Selection Algorithm.

Input: Q - Quality matrix for the set of network resources.
A - Quality requirements for a specified application.
Output: P - Optimal path for A.

```
1: function PCA_PS(Q, A)
2:     Q' ←normalize(Q, A)                    ▷ Normalize Q with respect to A
3:     R ← corr_matrix(Q')                    ▷ Calculate correlation matrix
4:     eigenvalues, eigenvectors ←det(R)
5:     eigenvalues ← sort(eigenvalues)
6:     CR, CCR ←calculate_cr_ccr(eigenvalues)
7:     loadings ←normalize_ev(eigenvectors)
8:     m ← 0
9:     while CCR[m] <= θ [1] do
10:        m ← m + 1
11:    scores ← [ ]
12:    for i = 0 to len(Q') do
13:        Y ← [ ]
14:        for j = 0 to m + 1 do
15:            y ← [ ]
16:            for k = 0 to len(Q'[i]) do
17:                y[k] ← loadings[j][k] · Q'[i][k]
18:            Y[j] ← CR[j] · sum(y)
19:        scores[i] ← sum(Y)
20:    P ← max(scores)
21:    return P
```

As discussed in Section 4.4, the principal components that account for 85%-95% of the total variance are carried forward, this is represented by the variable θ.

Python development tools, it is possible to embed Python within the User Policy Framework's C selection interface. In order to use a Python function within the C-based User Policy Framework, it is necessary to create an interface using the Python development headers, turning the inputs of the algorithm into C structures that Python can interpret and convert into objects. The real implementation for the PCA selection algorithm is found in pca_ps.py. The presented algorithm uses the NumPy library [111], to calculate the correlation matrix R and subsequently calculate and normalize the eigenvalues and eigenvectors, providing a matrix of loadings that can be used to compute the optimal service. The interaction between the presented PCA path selection algorithm and the rest of the User Policy Framework is presented in Fig.5.7, in the previous section.

5.5 Summary

In this chapter we provided an in depth description of the main components of both MAPD and the User Policy Framework. The MAPD was presented first, detailing the key components and their interactions, including the resource manager, network interface, interface handler and aggregation logic. These components come together to form a realisation of MAP presented in Chapter 4. Subsequently, we presented the User Policy Framework, providing a detailed discussion regarding management of the device and application contexts, specifying how additional contexts can be easily introduced into the framework. Furthermore the User Policy Framework required changes to the MPTCP implementation to support a user space controller, to interact with the kernel space path manager. Finally, the PCA based path selection algorithm was realized by integrating a Python module into the C-based User Policy Framework. The available Python interface will allow for additional path selection algorithms to be introduced easily.

CHAPTER 6

Evaluation

In this chapter, the initial focus is on evaluating the potential and overall improvement in network resource utilisation that can be achieved through the combination of MAP and MPTCP. Subsequently we will demonstrate the potential to further increase the users quality of service and experience through policy based network resource management and path selection. We break the evaluation down into seven components; first we investigate the efficacy of Multipath-TCP, observing how the Linux implementation performs with an increasing number of subflows, across a set of proposed congestion control algorithms include, cubic [69], LIA [135], and OLIA [99], we compare this to the current stock TCP implementation, establishing a standard TCP flow for each available path (6.1).

Following on from the initial evaluation of MPTCP, we begin to investigate MAP and its associated behaviours within the network. We use NS3 with the Direct Code Execution to run our real world implementation of MAPD inside a simulation environment. The simulation is required to observe the time taken and measure the overhead incurred by MAP as the size of the network increases (6.2). Furthermore, we present real world results from the perspective of a single device, investigating the time it takes to process and send an increasing amount of network resource. We then examine the cost of running MAPD on a host, with respect to the impact on battery life (6.2.3). Given the initial investigation of the MAPs overheads, we introduce active TCP traffic into the simulation environment, comparing the performance of MAP to traditional single-path and multipath approaches. Introducing traffic helps to demonstrate how the advertisement of additional network resource can benefit the user, by load balancing to spread traffic and in the case of MPTCP, sharing the available network re-

Protocol	Congestion Control	Flows
TCP	CUBIC	1-63
MPTCP	CUBIC	1-63
MPTCP	LIA	1-63
MPTCP	OLIA	1-63

Table 6.1: Experimental configurations for testing the performance and throughput of TCP and MPTCP with a variety of congestion control algorithms.

source more fairly (6.3). Furthermore, we present a realisation of this experiment on a physical testbed, using a combination of Raspberry Pi's and OpenWRT routers, further corroborating the benefits of our MAP approach (6.4). The final investigation of the MAP implementation relies on observing and evaluating the mobility properties. The mobility evaluation looks to demonstrate how MAP can improve resilience and reduce handover time by giving the hosts access to additional network resource that would not be traditionally available (6.5).

Continuing from the evaluation of MAP, we introduce the User Policy Framework splitting the evaluation of the work into two components. First we demonstrate how the resource management aspects can provide a fine grained approach to controlling and allocating network resource based on the users demands (6.6). We follow on from the proof of concept in resource management, to demonstrate how the proposed path selection algorithm can be used to improve the quality of experience received across a range of applications (6.7).

6.1 Multipath-TCP

Multipath-TCP was initially described in Chapter 2, the use of such protocols has had a significant impact on the design and implementation of MAP. To this end, we envisage the combination of MPTCP and MAP as one of the key usage scenarios, allowing the announced network resource to make effective use of resource pooling and mobility techniques. The purpose of this experiment is to validate that the performance of MPTCP does not degrade as the number of subflows increases. To achieve this we use the MPTCP full mesh path manager, with the modifications proposed in Chapter 5 in order to use up to 63 subflows simultaneously.

Provider	Type	Catagory	Mb	RTT	STD	Min	Max
Three	3G	HSPA	3.23	423.240	224.801	323.140	1510.172
Three	3G	HSPA+	7.49	61.939	14.836	43.600	138.000
Three	4G	LTE	12.63	60.997	6.875	47.090	86.918
EE	3G	HSPA	1.47	613.094	440.792	358.000	2343.000
EE	3G	HSPA+	6.67	107.835	9.525	96.200	127.000
EE	4G	LTE	10.51	60.997	7.241	47.090	86.918
Eduroam	WiFi	802.11n	24.80	21.227	17.441	2.277	112.697
Openzone	WiFi	802.11n	18.22	43.751	29.257	25.431	171.549

Table 6.2: Captured real world network latency, used with NetEm and NS3 to emulate or simulate real packet re-ordering across different MPTCP subflows.

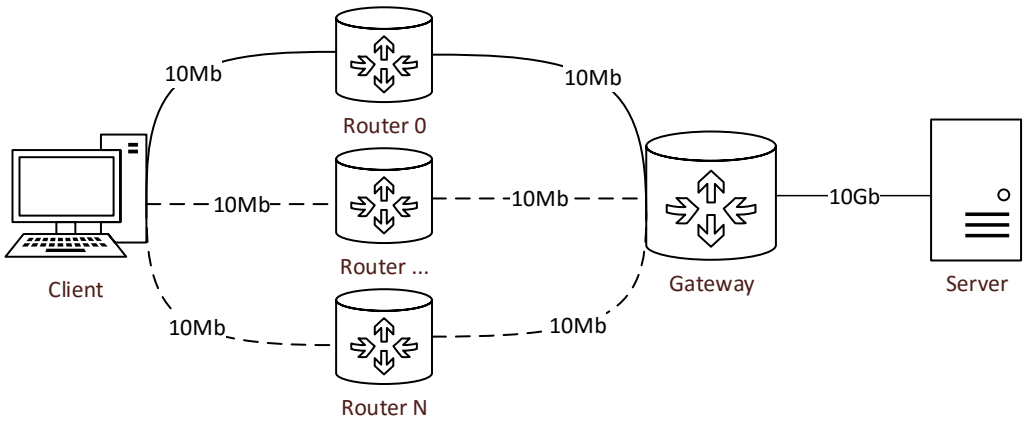


Figure 6.1: Topology for testing the potential throughput for increasing numbers of MPTCP subflows.

6.1.1 Experiment

In Table 6.1 on page 160 we present the base set of configuration parameters for the investigation of the scalability of MPTCP. For each combination of TCP or MPTCP and the presented congestion control algorithm, we examine the throughput achieved by establishing between 1 and 63 TCP flows, or MPTCP subflows, each with its own dedicated link. The proposed topology for this experiment is presented in Fig. 6.1 on the previous page, this figure shows a client which initiates the TCP connection and sends bulk data to the server using iPerf [75], the connection remains open for 60 seconds attempting to fill each of the network links. Each of the network links connects to a router from 0 to n, which are then aggregated at the gateway router. The gateway router, provides the server with a single network interface, fixing the number of subflows that are created to be equal to the number of network interfaces available on the client device. The bandwidth between the gateway router and the server is sufficiently large to prevent congestion, therefore a bottleneck is not introduced.

For scalability reasons, this evaluation is performed in NS3 with DCE, allowing the real Linux MPTCP kernel module to be used. In the case of TCP, we use iPerf, creating a number of parallel single-path TCP flows equal to the number of gateway routers in the simulation topology. While for MPTCP, iPerf creates a single TCP connection, allowing MPTCP to create the appropriate number of subflows (one for each network interface). Regarding the configuration of the MPTCP environment, we increase the TCP buffer sizes on each node to ensure that there is enough space to aggregate up to 63 subflows. Furthermore, we only consider the default MPTCP scheduler (lowest RTT first), which allocates packets to paths based on latency and available buffer space.

To improve the external validity of the experiment, we introduce variable network metrics. Firstly we include variable latencies, each of the network links between the client and the router delays packets according to the network metrics presented in Table 6.2 on the previous page, with a normal distribution related to the standard deviation. Subsequently we observe links with heterogeneous bandwidths, again using the network metrics we obtained in Table 6.2 on the previous page. Finally, we combine both the latency and bandwidth heterogeneity metrics, repeating the described experiment.

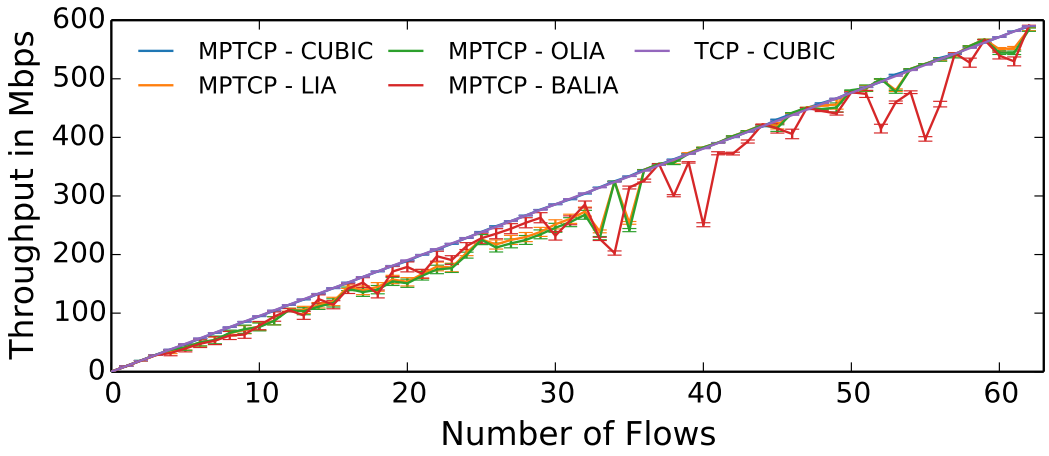


Figure 6.2: Throughput for TCP and MPTCP with an increasing number of flows, each transmitted over an independent path.

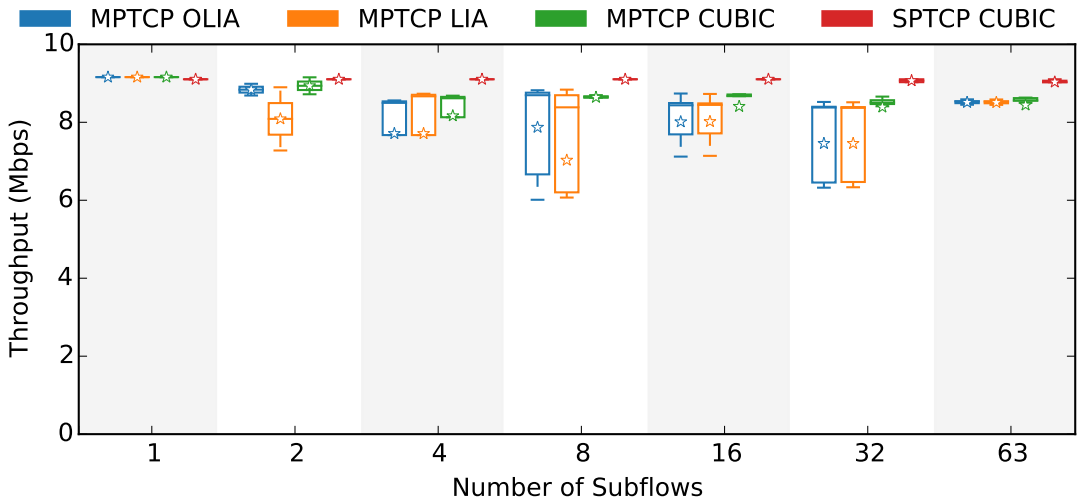


Figure 6.3: Utilisation of an increasing number of homogeneous paths.

6.1.2 Homogenous Results

The initial MPTCP scalability results presented in Fig. 6.2 on the previous page and Fig. 6.3 on the previous page show a promising foundation for the protocol; however, initial observations demonstrate that MPTCP does not perform as well as an equivalent single-path TCP approach in regards to link utilisation. In Fig. 6.2 on the previous page, both MPTCP and TCP with the cubic congestion control algorithm, scale linearly with the number of available homogeneous links. Despite this, MPTCP slightly under-performs the traditional TCP, by a small percentage. This similarity in performance provides a promising baseline for the performance of the MPTCP implementation at scale, however, this does assume that the TCP parameters at the host are tuned appropriately, ensuring the buffers are large enough to manage a large number of subflows. Additionally, Fig. 6.2 on the previous page demonstrates that the coupled congestion control algorithms (LIA and OLIA) are not yet suited to a vast increase in the number of subflows that are included in the shared calculation. Firstly this can be observed by the haphazard variability in throughput for both LIA and OLIA in Fig. 6.2. When the number of subflows established is approximately greater than 32, the congestion control algorithms become increasingly prone to integer overflow, this prevents the coupling of flows, causing the algorithm to fall back to uncoupled reno-like behaviour.

The integer overflow in question occurs due to the shared nature of the coupled congestion control approaches. As discussed in Section. 2.2, the congestion window is summed for each of the subflows, as shown in Eq. 6.1 and discussed in more detail in [135]. When the number of subflows associated with an MPTCP connection increases, the likelihood of overflow increases. In addition to overflow, this can also lead to imprecision in the calculations, which is amplified as the number of subflows increases. This is also observable in Fig. 6.3 on the previous page, when looking at the utilisation of 63 subflows, as the variance in throughput is greatly reduced. The fall-back to an uncoupled new reno, causes the throughput of each subflow to increase, however the benefits of a semi-coupled approach to congestion control, including balanced congestion and fairness at shared bottlenecks will no longer function as expected. For a single TCP flow, the performance

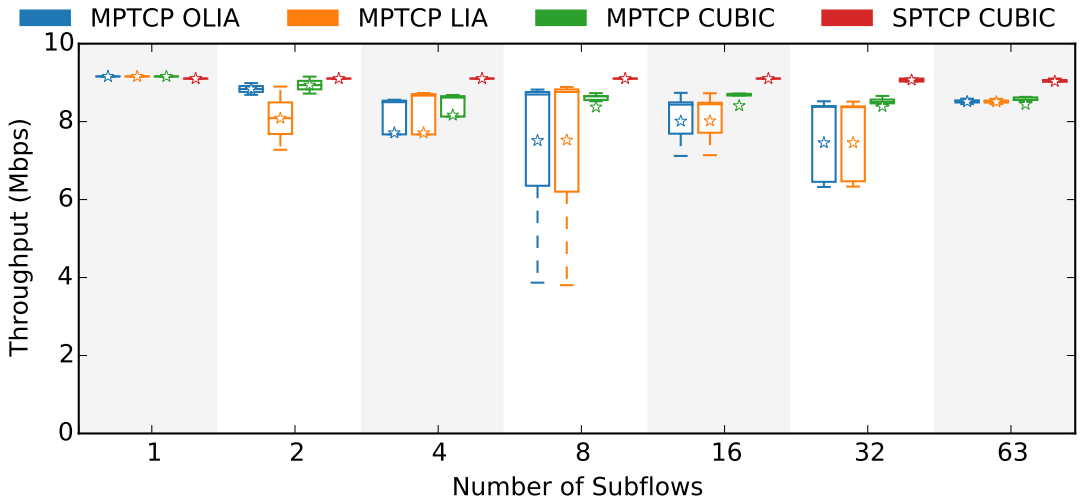


Figure 6.4: Utilisation of an increasing number of paths with heterogeneous latency.

of all congestion control algorithms are comparable.

$$total\_cwnd = \left(\sum_{i=1}^{i} \frac{{snd\_cwnd_i}^{\alpha} \cdot best\_rtt}{srtt_i} \right)^2 \tag{6.1}$$

6.1.3 Heterogenous Results

From the results presented in Fig. 6.4, it becomes apparent that there is little impact from introducing heterogeneous latency across the available links, in comparison to Fig. 6.3 on page 163. When using the default MPTCP scheduler, the subflow with the lowest round trip time fills its congestion window first, subsequently the subflow with the next highest RTT is used. When heterogeneous bandwidths are used however, the change in total utilisation is much more significant as presented in Fig. 6.5 on the next page. When comparing heterogeneous bandwidths to both bandwidth and delay heterogeneity, shown in Fig. 6.6 on the next page, the impact is as negligible as the introduction of variable delay in the homogeneous experiment. The impact of bandwidth heterogeneity degrades the overall utilisation relative to the number of network interfaces that are used. Despite the lower level of utilisation, MPTCP with CUBIC and either OLIA or LIA still performs well for the number of flows created, and additionally provides

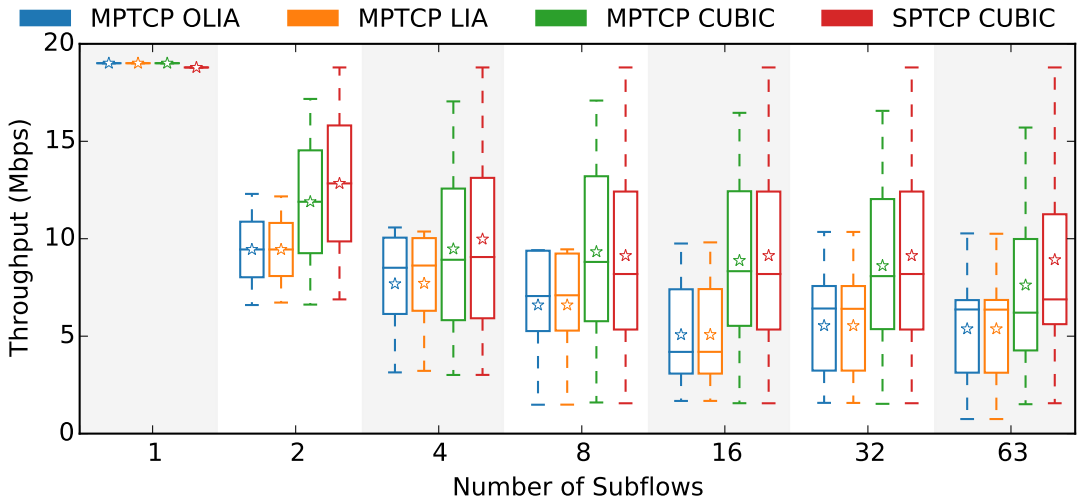


Figure 6.5: Utilisation of an increasing number of paths with heterogeneous bandwidths.

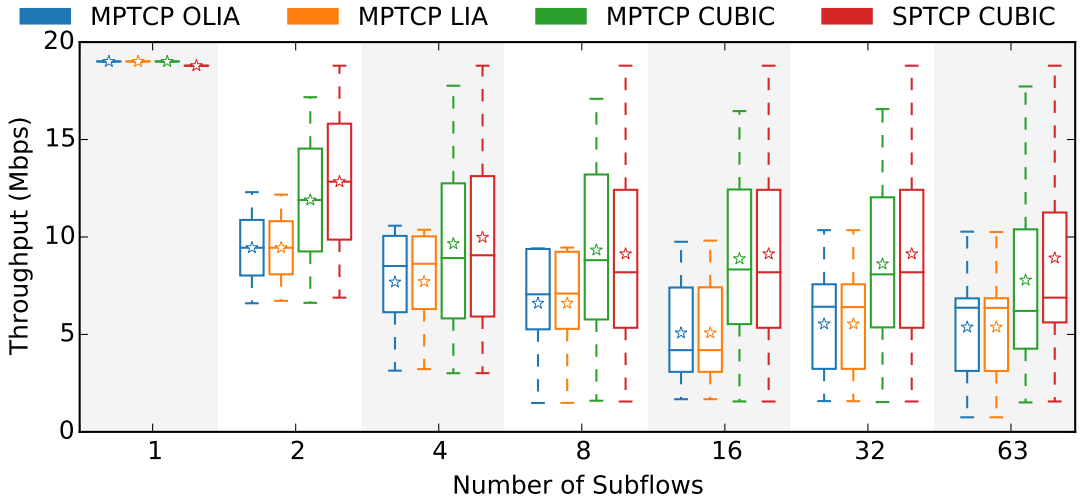


Figure 6.6: Utilisation of an increasing number of paths with heterogeneous bandwidths and latencies.

more flexibility than is available by default with TCP. Furthermore, iPerf and other application layer software, is typically not built to support such a connectivity model, therefore, the MPTCP based approach to utilizing the available links is much more transferable and adaptable to a real world context without modifications to the application.

Another important factor to note when looking at the result of combining both heterogeneous latencies and bandwidths, is the impact of the bandwidth-delay product on the TCP flows and their ability to increase their congestion windows accordingly. According to the metrics presented in Table. 6.2 some of the links have significantly large round trip times, in excess of 400 milliseconds on average. If the round trip time is relatively large, as is the case in some of the cellular networks, then the 60 seconds allocated for the experiment may not be sufficient to provide an accurate measure of the available bandwidth. This is due to the flow spending excessive amounts of time in the slow start phase, which is designed to quickly estimate the available bandwidth in the network, before moving into the congestion avoidance phase. For a new TCP flow, an initial conservative congestion window size is chosen typically consisting of 2, 4 or 10 segments [26], this is doubled for each round trip time while in the Slow Start phase. Therefore, in the case of the worst performing network path (EE over HSPA) with a RTT of 613 milliseconds and a 1.47 Mb bandwidth, it will take approximately 4.5 seconds to reach a congestion window size of 923KB based on the optimal number of in-flight packets. Given the variable latency that has been introduced into each path, in the best case scenario, the poorest path will therefore spend at most 55.5 seconds in the congestion avoidance phase. In the worst case scenario for the proposed experiments, the initial slow start phase is estimated to take up to 8.5 seconds, which will still allow the poorest flows to spend a sufficient amount of time in the congestion avoidance phase.

6.2 MAP Behaviour

For the evaluation of MAP, we first consider the impact that the protocol has on a variety of different network topologies. The different network topologies we will investigate are derived from a tree, such that there is always a root node aiming to disseminate its information, to all the child nodes. In the NS3 environment, we

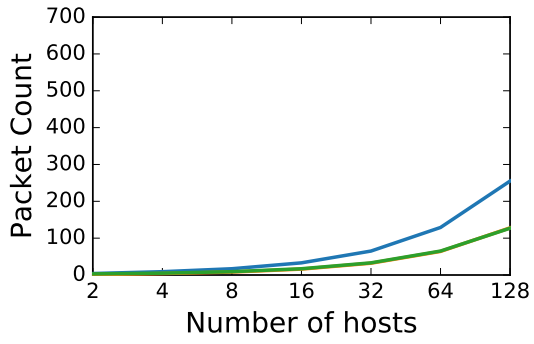
Exp Name	Description	Nodes	Stride	Gateways
(a)	Flat Network	1-128	128	1
(b)	Nested Network	1-128	1	1
(c)	Flat Network	128	128	1-128
(d)	Nested Network	128	1	1-128
(e)	Variable Stride	128	1-128	128

Table 6.3: Network Topologies with an increasing number of Internet enabled gateways, installed at the root node.

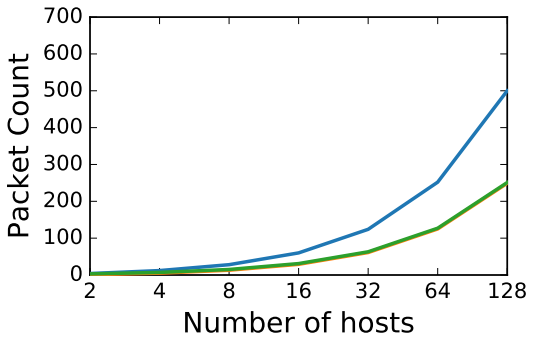
focus on devices with local WiFi connectivity between one another. There are two key network metrics to investigate, the latency and the overhead. In this context, latency refers to the total time taken for each node in the network to receive and process the updates provided by the root node, referred to as convergence time. The overhead is measured by observing the number of bytes and the number of packets transmitted, for the network to converge, this is broken down into the request and update packets that each host sends. Furthermore, we additionally investigate the impact that MAPD has on a single host, with respect to power consumption and the time required to process updates.

6.2.1 Overhead Results

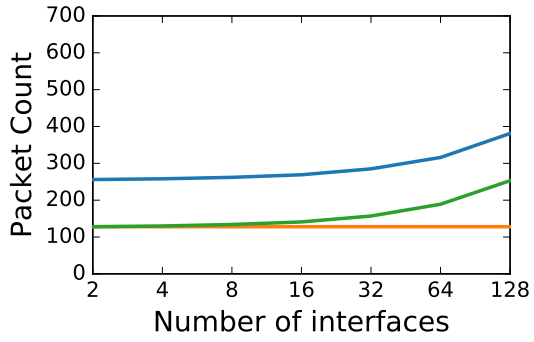
In this section, we present the changes in overhead that have been obtained during the presented behaviour experiment. Fig. 6.7 on the next page demonstrates that the number of packets required to advertise network resource scales linearly with respect to the number of hosts in the network. This is evident from Fig. 6.7a on the next page and Fig. 6.7a on the next page, while the graph is curved the number of hosts represented on the x-axis increased exponentially, resulting in a linear relationship. In Fig. 6.7a on the next page, we present a flat network with all hosts sharing a single link, the number of request packets is equal to the number of hosts in the network, while the number of updates is approximately doubled. In Fig. 6.7b on the next page, the number of requests is equal to the number of network interfaces in the topology, as each host broadcasts on each of its connected subnets. The number of updates transmitted also corresponds to the number of network interfaces, as opposed to the number of hosts, this further corroborates the results in the flat network in Fig. 6.7a on the next page, as the number of hosts is always equal to the number of network interfaces. The rate of



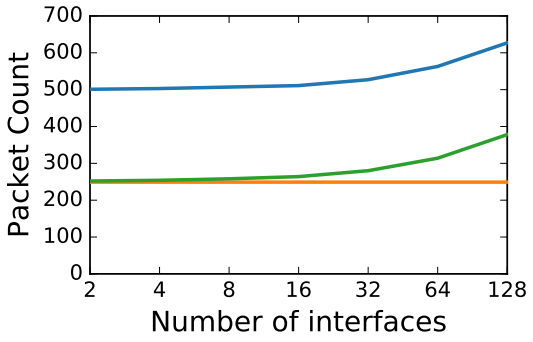
(a) Flat Network (gateways=1)



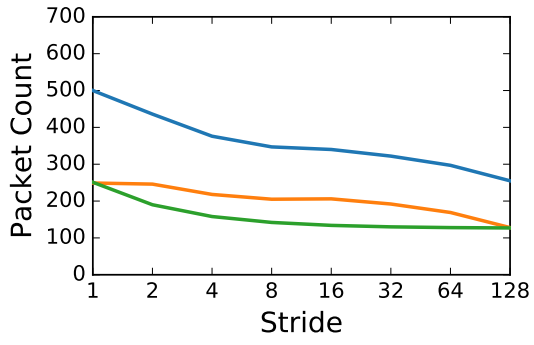
(b) Nested Network (gateways=1)

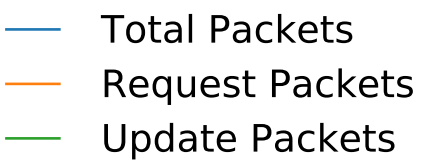


(c) Flat Network (hosts=128)



(d) Nested Network (hosts=128)





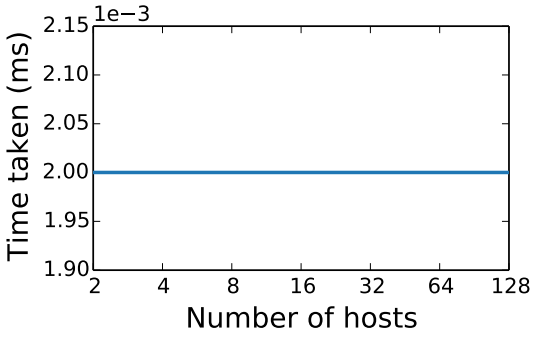
(e) Variable Stride (gateways=128, hosts=128)

Figure 6.7: Overhead of running MAPD on various network topologies.

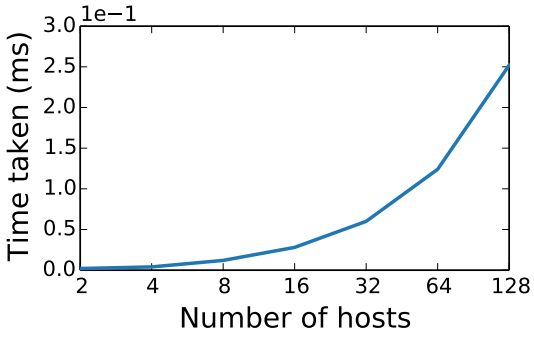
growth in terms of overhead shown in Fig. 6.7c on the previous page and Fig. 6.7d on the previous page, as the number of network interfaces rises is much lower in comparison to increasing the number of interior network interfaces (inferred by an increased number of hosts). By increasing the number of network resources advertised, the number of packets required to represent this information slowly increases. There are a number of reasons for such an increase, such as hitting the MTU of the network requiring multiple packets for a single update, which in turn may increase the number of collisions and subsequently retransmissions. This however does not account for the slow increase that is presented, which suggests improvements to MAPD could be made. Finally we present the variable stride topology in Fig. 6.7e on the previous page, this figure effectively demonstrates that the number of hops in the network (again, interior network interfaces) has a more significant role than the number of advertised network resources, but is secondary to the number of network interfaces due to the slower growth, in comparison to Fig. 6.7a on the previous page and Fig. 6.7b on the previous page.

6.2.2 Latency Results

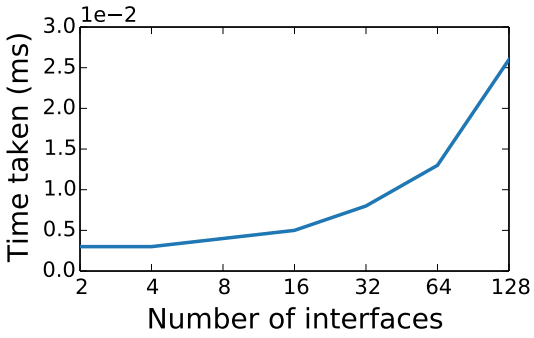
In Fig. 6.8 on the next page we present the results of measuring total convergence time for varying networks. In Fig. 6.8a on the next page the stride of the tree is larger than the maximum number of hosts, leading to a single subnet being created for all communications. To this end, the time taken to converge is constant no matter how many hosts are introduced into the network. It is expected however that a significant raise in the number of hosts would slowly start to increase the time required for a flat network, due to packet collisions. In the case of Fig. 6.8b on the next page, including both transmission and processing, the time required scales with the number of hops in the network taking approximately 2ms per host, with no interference or cross traffic in the WiFi environment. When the amount of announced network resource is increased in Fig. 6.8c and Fig. 6.8d on the next page, there is still a linear increase for a constant number of hosts. As the amount of network resource increases, the amount of time to process and transmit grows. Firstly the number of packets required to transfer the entire MAP update increases as the MTU is reached, secondly each host must spend more time processing the update, creating the appropriate address, routes and rules. As expected, the impact of increasing the number of announced network



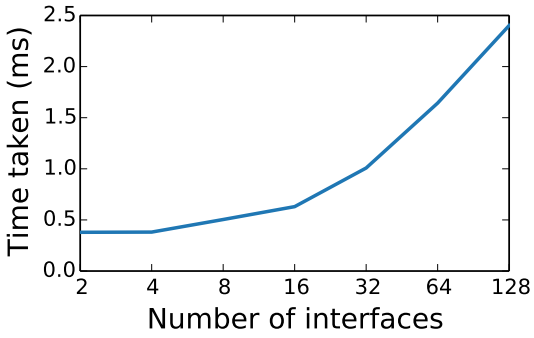
(a) Flat Network (gateways=1)



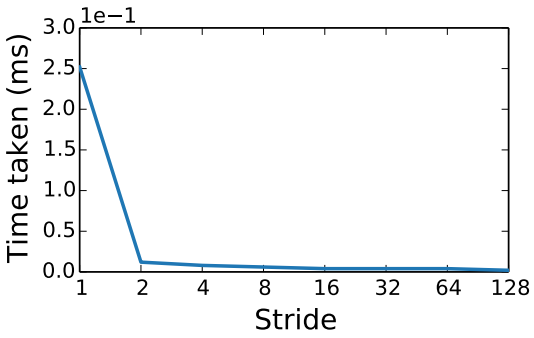
(b) Nested Network (gateways=1)



(c) Flat Network (hosts=128)



(d) Nested Network (hosts=128)



(e) Variable Stride (gateways=128, hosts=128)

Figure 6.8: Time taken for dissemination of MAPD to complete for various network topologies.

Resources	1	2	4	8	16	32	64
Time	0.005	0.014	0.023	0.048	0.103	0.363	1.117

Table 6.4: Time taken in seconds for MAP to process an increasing number of network resources from a single update.

HBPM	240	120	60	30	15	0	Base
Send	30.30	30.17	30.08	29.96	29.97	29.89	22.84
Recv	30.58	30.67	31.88	31.73	31.16	29.90	22.84

Table 6.5: Energy (Joules) expended by a Raspberry Pi 2 with WiFi connectivity, at different frequencies of heart beats per minute (HPBM). The base value represents a device with no network connectivity.

resources has a more significant impact on the nested topology as opposed to the flat topology. The convergence time for the variable stride topology is shown in Fig. 6.8e on the previous page, we begin with the smallest stride of one, and increase at a rate of 2^n, as expected at each incremental value of n the number of hops decreases significantly, leading to a large drop in convergence time.

6.2.3 Device Impact

When measuring the impact that MAPD has on a device, we focus our experiments on the Raspberry Pi 2. Our first local view of MAPD, looks at how long it takes the MAP implementation to process varying sizes of network updates. For this experiment, shown in Table 6.4, we consider a single host that receives a network update, with a pre-determined number of advertised network resources. The time taken for each number of announced resources is measured ten times and the mean value is presented. As we verify both additions and deletions of network resource against the current state, we incur a computational complexity of $O(n^2)$, which can be seen in Table 6.4. It is likely that in most real world mobile networks, the number of advertised links would remain fairly small, limiting the computational impact of the current algorithm. Furthermore, as the resource advertisements are processed sequentially, even with a significantly larger number of network resources being announced, the host will be able to use each successive resource as soon as the individual entry in the advertisement has been handled.

The second device specific experiment presents the power consumption implications for transmitting the additional MAP packets, which are necessary to

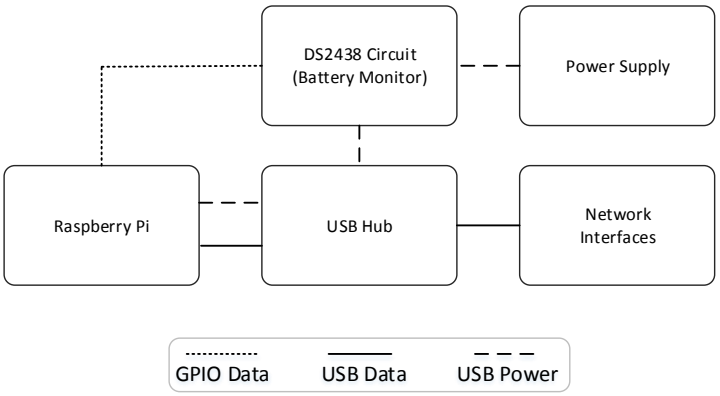


Figure 6.9: Topology for testing the potential throughput for increasing numbers of MPTCP subflows.

maintain the appropriate routing tables to allow mobile devices to use any available Internet connection. The setup and topology of this experiment is the same as the previous experiment; a single Raspberry Pi 2 is used, with a 2.4GHZ 802.11n USB WiFi adapter (TP-Link WN722N). A controlled MAP host is then used to automatically send updates at the specified rates, and alternatively receive them when the Raspberry Pi is transmitting its own information. The power consumption is averaged using the mean value over the course of five minutes, and repeated ten times for each frequency of heartbeat. To obtain the information regarding current draw and voltage, we use a bespoke monitoring solution using the DS2438 Smart Battery Monitor [82], the setup of which is shown in Fig. 6.9. The results of the power consumption experiment are shown in Table 6.5. This table, shows the energy expended in Joules, for different frequencies of heart beats per minute (HBPM), for both sending and receiving MAP updates. Receiving and processing updates is more computationally expensive, increasing the overhead on the CPU and slightly increasing the power consumption. At the higher frequencies (120 and 240 HBPM) a small percentage of updates were lost, reducing the CPU overhead and therefore slightly reducing the power consumption. Assuming a conservative 2000mAh battery at constant load, at the highest power draw, the user would lose approximately 12 minutes of battery life, from a total of fours hours if the device and wireless were idle while not running MAPD, as presented by the base value.

Scenario Name	Transport Protocol	Load Balancing	MAPD
SP	TCP	No	No
LB	TCP	Yes	No
LB MAP	TCP	Yes	Yes
MP	MPTCP	No	No
MP MAP	MPTCP	No	Yes

Table 6.6: TCP/MPTCP configurations used for each simulation scenario.

6.3 MAP Network Utilisation

In this section of the evaluation we will focus on the throughput that a mobile network is able to achieve given a set of different approaches to Internet connectivity, ultimately focusing on the benefits of MAP. To this end, there are five key scenarios that are accounted for, which are shown in Table 6.6. Firstly we observe the behaviour of standard TCP (**SP**) in the proposed network topologies. The case of TCP operating on its own is considered to be the current default state, that we aim to improve on. The second scenario (**LB**) incorporates load balanced routing in each of the hosts, still running standard TCP, this is the optimal case for a single host, that is not currently supported by default in mobile devices and requires advanced configuration in Linux operating systems. In this experiment, load balancing refers to hosts spreading their traffic across the set of available network resources, on a per flow basis. The third scenario (**LB MAP**) combines TCP with both load balancing and MAP. In this case the MAPD implementation is responsible for the configuration of load balancing as opposed to being statically configured in the simulation; indirect connectivity is advertised as described in Chapter 5. Therefore the load balancing achieved spreads traffic across the local and remote external connectivity, as additional network resource has been advertised by the MAP. The fourth scenario (**MP**) focuses on MPTCP, the necessary routing and rules are configured manually in the simulation configuration, allowing each host to use each of the directly connected links incorporating them into an MPTCP connection. Finally in the last scenario (**MP MAP**), MPTCP is combined with MAP, such that subflows are created for each of the known Internet connections whether they are local or remote, providing cooperative resource pooling throughout the mobile network. As in Section 6.1 we tune the TCP buffers to ensure that they are large enough to

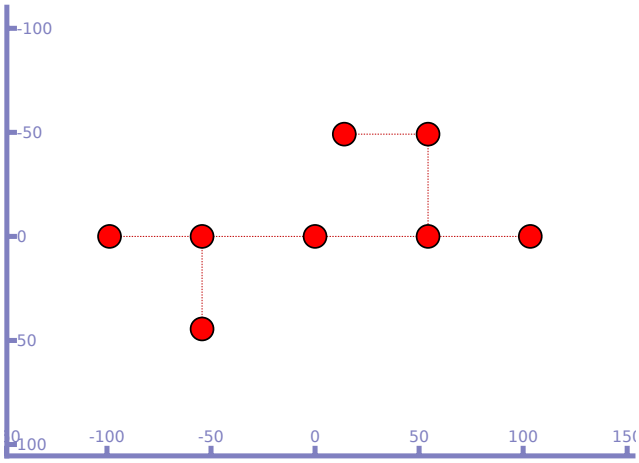


Figure 6.10: Visulisation of nested tree topology for MAP simulations.

handle aggregating multiple MPTCP subflows. Additionally, in this section we exclusively use CUBIC, as it represents the closest match between single-path TCP and MPTCP, as demonstrated in Section 6.1.

In this experiment, the base singlepath TCP (**SP**) scenario considers the current use case when two or more hosts form a mobile network. In this context, a single default route is used remaining constant for each connection. The Load Balanced scenario (**LB**), refers to the closest current optimal real world implementation, in which the hosts are able to load balance over their available network connections. This scenario represents a mobile network, which may be supported by Ad-Hoc protocols such as BATMAN [1] or OLSR and the hosts are configured to load balance across the set of locally available network interfaces . Subsequently the addition of MAP (**LB MAP**) demonstrates the improvements that can be achieved through the advertisement of additional network resource, and therefore the benefits of sharing. This is augmented further when combining MPTCP with MAP (**MP MAP**), which aims to show how resource pooling can increase the utilization of the available network resource.

6.3.1 Topology

In the previous experiment in Section 6.2, describing the behaviour of the MAP protocol, we focused on two distinct network topologies, nested trees with mul-

[1] In the case of BATMAN, hosts are able to annouce their intention to act as a gateway, each host allocates a single egress network resource which may change over time, leading to a simple load balanced solution.

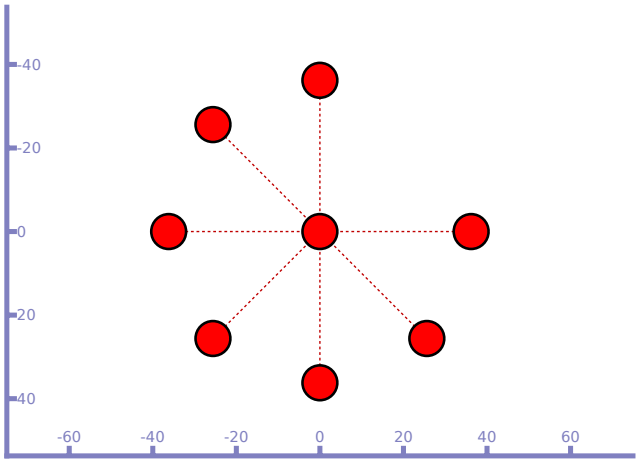


Figure 6.11: Visulisation of flat topology for MAP simulations.

tiple hops shown in Fig. 6.10 on the previous page, and a flat network with a single sharing link presented in Fig. 6.11. In terms of network connectivity, both of these network topologies have been defined and used in multiple contexts. The tree topology has been proposed by the MANEMO community [110], and has recently been used in the context of sharing network resource [4]. The flat "sharing link" approach was presented by Stiemerling et al. in [163]; which is used to cooperatively stream live P2P video and retrieve web content. One of the benefits of the MAPD implementation is the flexibility and configurability that is enabled regardless of the network topology or infrastructure that it is deployed upon. Each of these topologies has advantages and disadvantages in a mobile scenario; for example, the flat network may simplify routing, reducing overhead and convergence times as previously demonstrated. The tree (or mesh) based approach is more complex from a routing perspective, however it can also increase the range of connectivity and provide flexibility in terms of connectivity options, through multiple network access mediums, i.e. WLAN, Bluetooth, or USB.

Firstly we ensure that there is free network resource available, by limiting the number of hosts that generate TCP traffic to 50%. To create traffic, we use a patched iPerf3 for DCE, which allows multiple TCP connections to run sequentially without blocking. Each designated host creates two parallel threads, each of which creates ten flows sequentially of varying sizes including: 64KB, 512KB, 4MB and 32MB. A variety of flow sizes are necessary in order to demonstrate the behaviour of TCP and MPTCP with both mouse and elephant flows, better representing real world behaviour. The choice of sizes was informed by previous

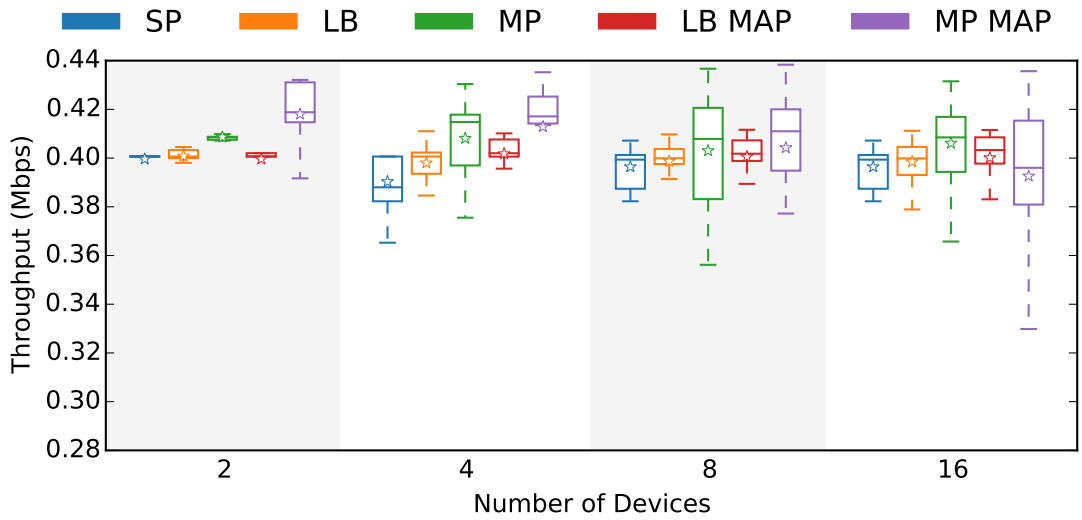


Figure 6.12: Throughput from simulation of the tree based topology for **64KB** flows using iPerf.

work on MPTCP by Paasch in [121]. Furthermore, within each of the discussed scenarios, we test small to medium size networks ranging between two and sixteen hosts. The upper bound in terms of network size is limited by the simulation environment. The local connectivity between hosts is limited to WLAN (802.11) as with the previous experiments, while the Internet connections are dedicated Point to Point links. Similarly to the previous MPTCP scalability experiment, each of the Point to Point Internet connections is shaped according to the metrics provided in Table 6.2 on page 161.

6.3.2 Results

In Figs. 6.12, 6.13, 6.14 and 6.15, we present the utilisation of the tree network for different resource sharing approaches for the discussed set of flow sizes. Across the presented figures, the primary observation is that the introduction of MAP performs better with an increased size of flow.

For the 64KB flows size shown in Fig. 6.12, MAP with MPTCP performs better than the alternatives when there are between 2 and 8 devices in the network. When the number of devices is increased to 16, the performance of MAP with MPTCP decreases by 3.3% in comparison to an independent MPTCP and a decrease of 0.76% in comparison to single-path TCP. For two devices, MAP with

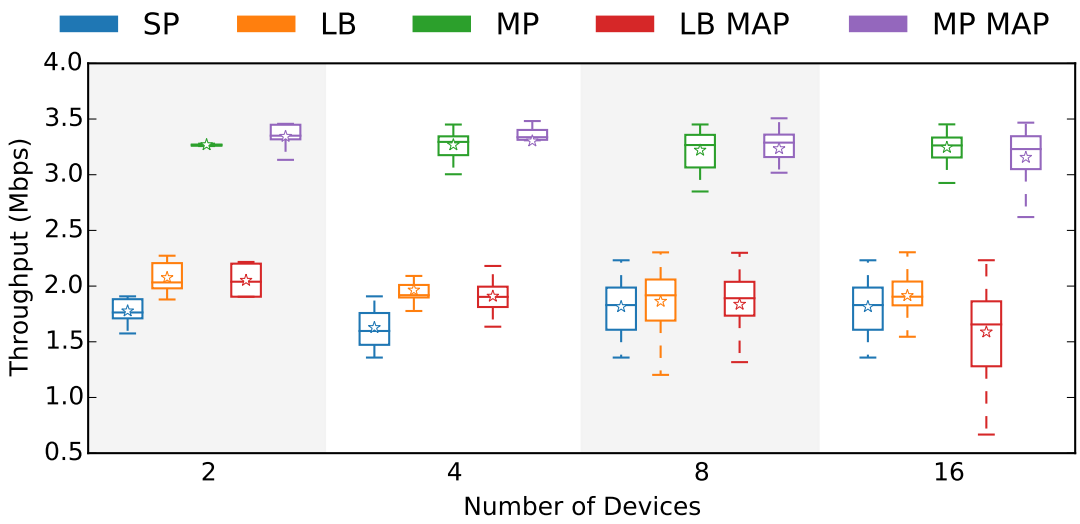


Figure 6.13: Throughput from simulation of the tree based topology for **512KB** flows using iPerf.

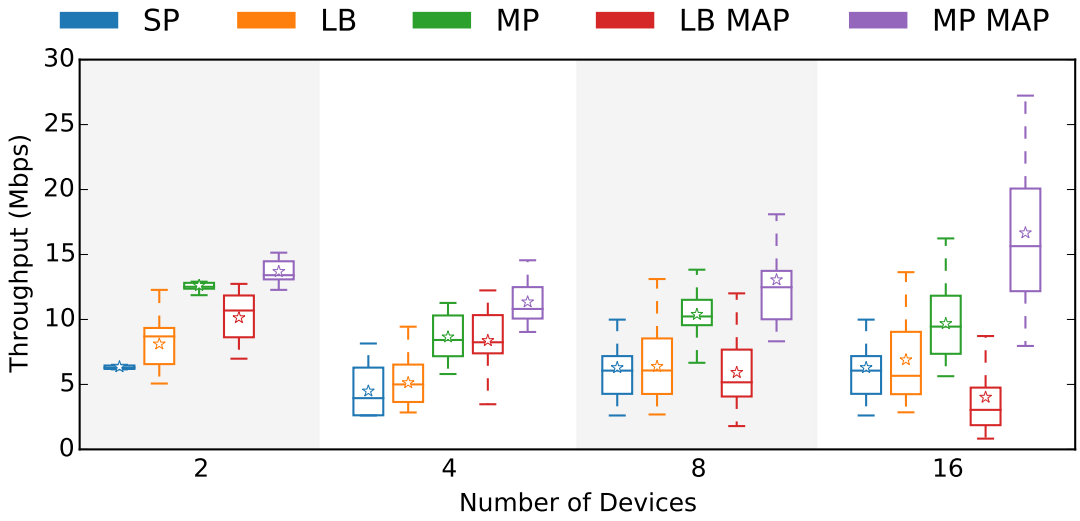


Figure 6.14: Throughput from simulation of the tree based topology for **4MB** flows using iPerf.

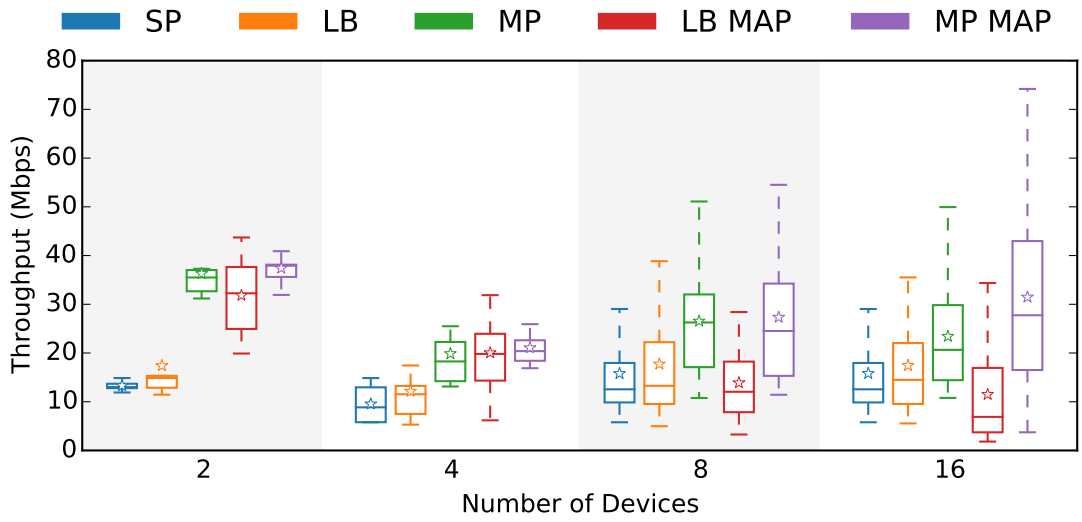


Figure 6.15: Throughput from simulation of the tree based topology for **32MB** flows using iPerf.

MPTCP shows a small improvement of 2.2% over the standalone MPTCP. In the case of the 512KB flows shown in Fig. 6.13 on the previous page, the performance benefit models that of the 64KB experiment as there is a small benefit between 2 and 8 devices (ranging from 2.29% to 0.4%), with a slight decrease in performance for 16 devices of approximately 2.9%. While there is a small decrease in performance under MAP when considering 16 devices for 64KB and 512KB flows, this provides a strong argument for the proposed User Policy Framework combined with the user space MPTCP controller, as subsets of high quality paths could be selected, ensuring that MPTCP with MAP always outperforms standalone MPTCP or single-path TCP.

When the file size is increased to 4MB as illustrated in Fig. 6.14 on the previous page, the benefits of using MAP with MPTCP become evident. Considering the mean value, for each number of devices tested, MAP and MPTCP outperforms the independent MPTCP test by 8.5% for 2 devices, 31.2% for 4, 25.6% for 8, and finally by 72% for 16 devices; this presents a phenomenal improvement in performance compared to the 64KB and 512KB experiments. Observing the 32MB flows, the mean improvement of MPTCP with MAP compared to pure MPTCP, is 17.5%, 5.1%, 3.7%, and 34.2%, with respect to 2, 4, 8, and 16 devices. While these performance benefits are not as significant as demonstrated by the 4MB flow experiment, this may be an artifact of the decrease in MPTCP

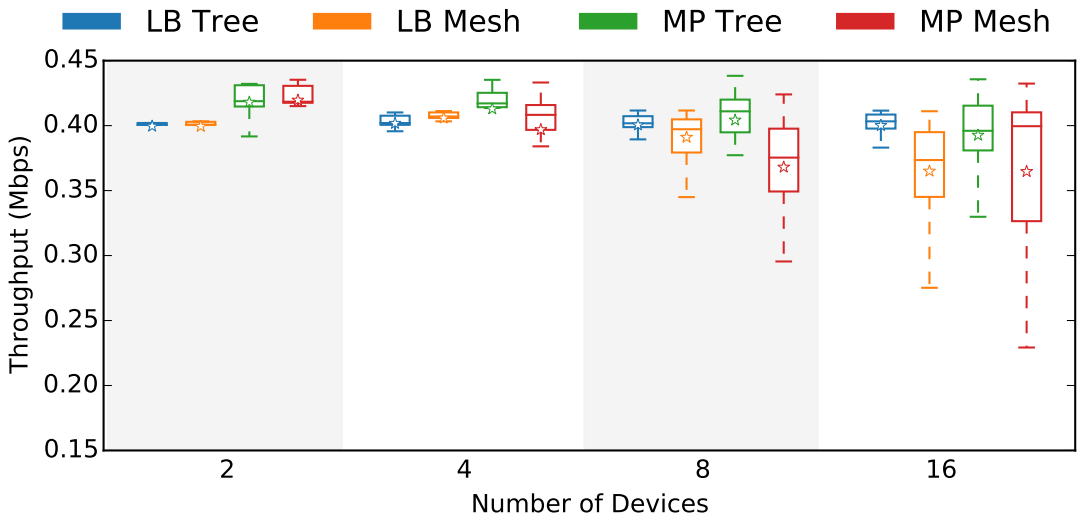


Figure 6.16: Throughput from simulation of the mesh based tree topology for **64KB** flows using iPerf.

performance in a heterogeneous environment as shown in Section 6.1.

In regard to MAP with load balancing, the measured utilisation is generally poor across all the simulated experiments, in comparison to the alternative connectivity approaches. There are two factors attributing to this problem, first the load balancing approach is a simple round robin, so slow links will become heavily congested as flows are added to them. Secondly, due to the nature of the topology, using MAP ensures that all the poor quality links and paths will also be used, degrading the overall throughput per host. This is exacerbated as load balancing without MAP ignores a number of interfaces, and due to the topology and distribution of links and flows, leads to it favouring the higher quality links.

Enforcing the tree topology with the MAP protocol, appears to be detrimental and does not maintain a fair and altruistic approach to accessing network resource; each additional hop away from the root node increases the total amount of network resource available to a given host. For example, in our topology each host has its own Internet capable gateway, therefore the root node will only have access to its own resource, while sharing it with all child nodes. This could vastly hamper the incentive for users to share their connectivity, and provides a justification for the MAP to break the tree model. Disseminating the network resource advertisements downwards as well as up towards the root of the tree, gives each host in the network equal access to the available network resource.

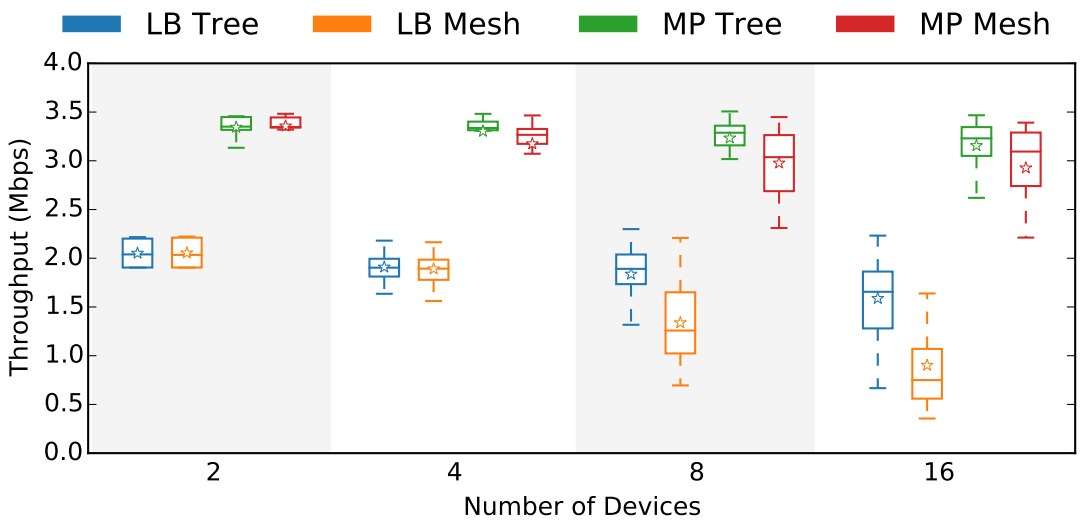


Figure 6.17: Throughput from simulation of the mesh based tree topology for **512KB** flows using iPerf.

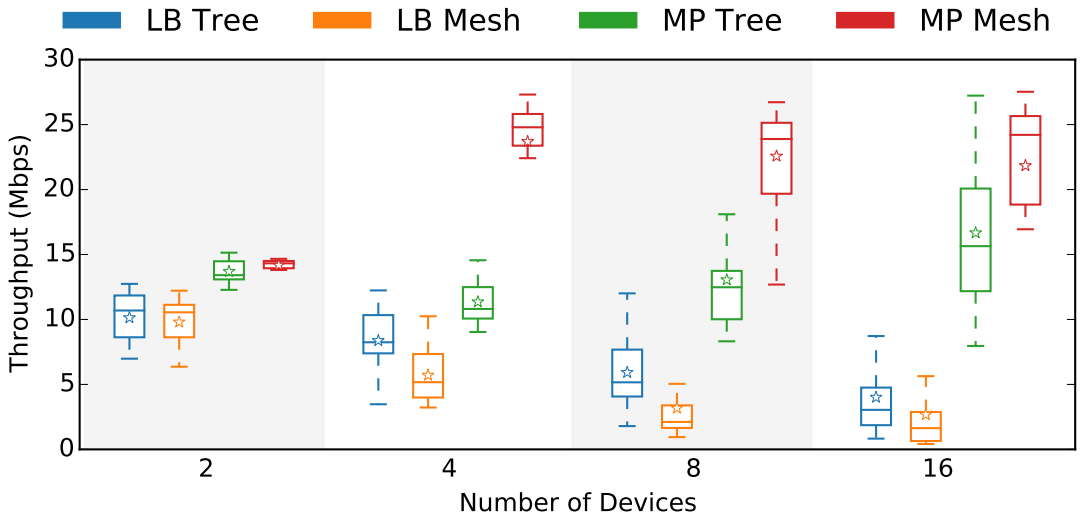


Figure 6.18: Throughput from simulation of the mesh based tree topology for **4MB** flows using iPerf.

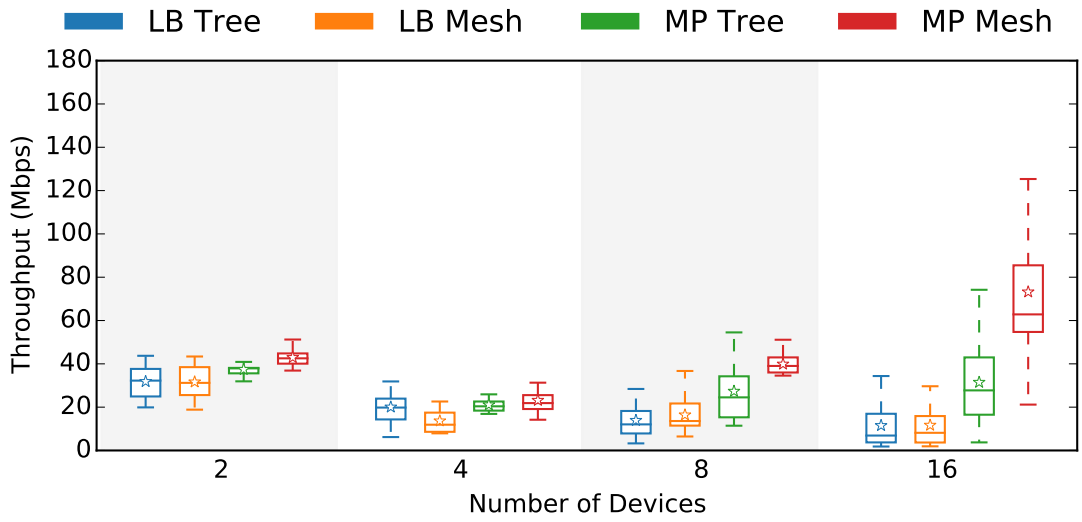


Figure 6.19: Throughput from simulation of the mesh based tree topology for **32MB** flows using iPerf.

Due to our observations regarding the state of altruism in the tree topology, we repeated the previous experiments comparing the traditional tree in which network resource is exclusively pushed downwards, to a mesh model, in which the resource is pushed to all hosts. As forming a mesh with MAPD provides no benefits to the SPTCP, SPTCP-LB, or MPTCP modes of connectivity, they will not be included in the comparison. The graphs presented in Figs. 6.16, 6.17, 6.18 on the previous page, and 6.19, demonstrates that enabling MAPD to access any network resource in the tree topology, not only that belonging to the parents, amplifies the previous results. For example, in the case of small 64KB flows, increasing the number of network resources available decreases the mean performance by 7.1% for 16 devices. On the contrary, for larger flows including 4MB and 32MB, increasing the amount of network resources presents a large increase in the mean throughput achieved. Specifically in the case of testing 32MB flows, the mean throughput for 16 devices is increased from 31.47 Mbps to 73.192 Mbps, leading to a total improvement of 132%.

For the flat "sharing-link approach", the resulting graphs are provided in Appendix B.2 on page 242, demonstrating similar improvements and behaviour to the nested topology that has been presented. As the results from the sharing link experiments corroborate those of the tree, this presents additional proof that MAP is flexible and adaptable to variable topologies. Additionally, we include

Link	Type	Catagory	BW (Mbps)	RTT(MS)	Loss (%)	Jitter (MS)
L1	4G	LTE	7.15	141.62	0.20	91.83
L2	4G	LTE	10.72	65.84	0.03	14.94
L3	4G	LTE	15.63	124.64	0.05	50.51
L4	4G	WiMax	2.72	166.22	0.26	55.52
L5	WLAN	802.11	10.72	46.97	0.20	22.80
L6	4G	LTE	12.01	77.31	0.13	21.43
L7	3G	HSPA+	6.49	115.32	0.14	31.06
L8	3G	EVDO	1.19	328.02	0.27	151.28
L9	WLAN	802.11	18.32	39.92	0.25	16.82
L10	3G	eHRPD	1.85	362.37	0.10	135.28

Table 6.7: Emulated link characteristics applied to network links in real world deployment.

results for the nested topology, in which a single user initiates an iPerf session as opposed to a set of users. This demonstrates that when using MAP, as the amount of free network resource increases the net benefit to the user also increases.

6.4 MAP Real World

To evaluate the real world impact that our implementation has on the utilisation of the network, we consider the testbed setup shown in Fig. 6.20. This topology has been derived from the Fire Service use case presented in Chapter 1. The mobile network is established using a range of 2.4Ghz and 5Ghz WiFi. Each Internet connection in the mobile network is emulated based on a link in Table 6.7, which have been drawn from real world measurements of a set of heterogeneous access technologies in [24]. The difference in topology from the previously discussed simulations instantly provides an alternative perspective regarding the benefits of introducing MAP into a network. While the number of network interfaces is kept fairly small, at only five, the hosts do not have direct access to their own network connectivity, and are instead relying on access via the Ad-Hoc mesh network. Initially we look at the throughput achieved by each host, with and without MAPD running, presented in Table 6.8 on the next page. For each experiment, iPerf runs 5 times over 30 seconds and the results are averaged; furthermore, we compare each host establishing an independent MPTCP connection with each host competing for network resource simultaneously. The presented network has an aggregate mean throughput of 22.97Mbps, based on the links chosen from Ta-

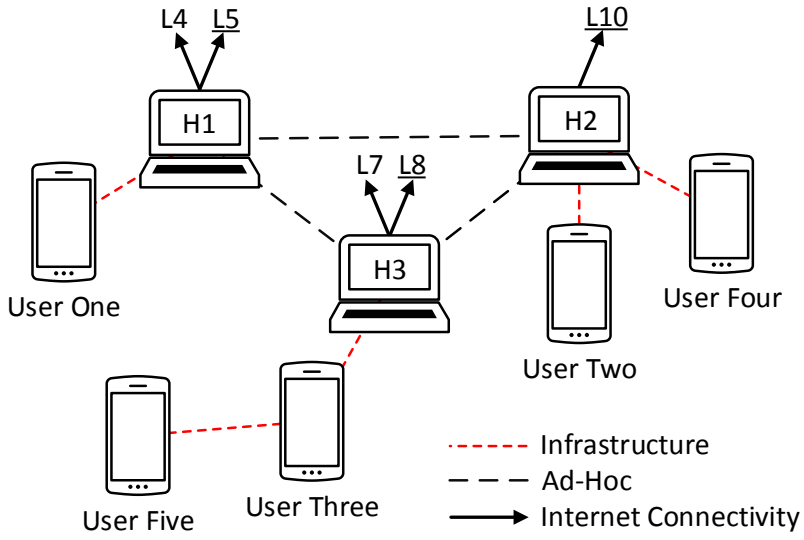


Figure 6.20: Real world network topology for evaluating the benefits of MAP.

	Single		Shared	
Host	**MP**	**MAP**	**MP**	**MAP**
U1	10.268	21.180	10.208	4.398
U2	1.736	14.948	1.466	2.920
U3	2.010	10.682	1.428	4.070
U4	1.746	15.640	1.462	3.846
U5	1.944	10.545	0.659	4.004
Total			15.228	19.238

Table 6.8: Host centric comparison of MPTCP with and without MAP in the presented real world network.

ble 6.7. When the hosts are competing for bandwidth, this leads to a utilisation of 66% without MAPD running, and 84% with MAPD. When each of the hosts has open access to all the network resource using MAPD, the total utilisation increases significantly compared to the MPTCP only approach, from 15.4% for an independent MPTCP to 63.5% using MAPD.

Following on from the initial perspective on utilisation with MAP, we repeat the previous NS3 simulations, analysing the behaviour for varying connectivity approaches. For the SPTCP approach, each host will simply use the default route of the gateway they are connected to, which has been underlined in Fig. 6.20. When load balancing, each of the gateways will spread flows across their own

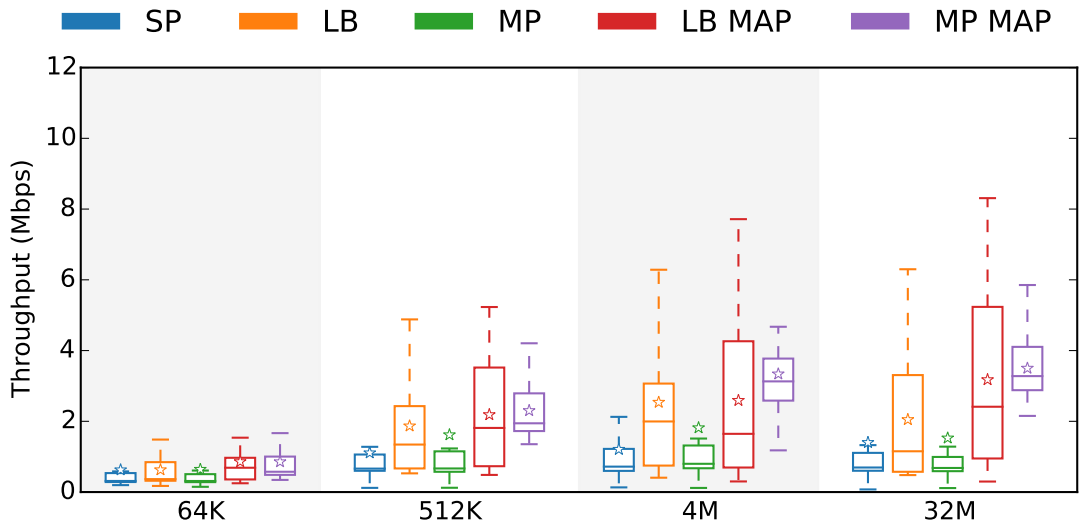


Figure 6.21: Throughput for the real world network, transferring a variety of file sizes.

network interfaces. For MPTCP, as each of the hosts initiating the connection do not possess multiple default routes, they will be unable to leverage the power of multipath. When introducing MAP into the network, all of the hosts and gateways are made aware of all the available Internet connectivity, increasing the amount of network resource that can be accessed by both the load balancing and MPTCP approaches. In keeping with the simulations, as before we observe TCP flows of the following sizes: 64KB, 512KB, 4MB and 32MB. From the results of this experiment, shown in Fig. 6.21, it is immediately obvious that the total network utilisation is inadequate when the hosts are unaware of the additional Internet connectivity that is available to them. Moreover, even without multipath resource pooling, allowing the hosts to load balance over all the available Internet connectivity improves utilisation in comparison to unaware load balancing of each host. Due to the topology of the network, host based resource pooling technologies provide no benefit to the traditional single-path approach. If the user devices were to be extended with their own Internet connectivity, while utilisation would increase it would be unlikely to surpass the presented load balancing approach, as indirect Internet connectivity is ignored. When load balancing with MAP, the total network resource utilisation was hampered as each host still attempted to use the poor quality links. This leads to the high capacity links being neglected as the hosts compete for bandwidth on the lower capacity links. Therefore, to

Time	Event	Hosts
0	Data Start	U1, U2, U3, U4, U5
40	Connection	H1, H3
70	Connection	H1, H2, H3
90	Connection	U6 (L1), H1
90	Data Start	U6
130	Link Down	H1 (L4)
150	Migrate	U4 (H2 to H1)
190	Link Up	H3 (L7)

Table 6.9: Description of mobility events, for users, hosts, and links.

optimally utilize the available connectivity in a mobile network, combining MAP with MPTCP provides the best solution. This is evident as the benefit of load balancing both with and without MAP will always be dependent on the number of active flows and the ability to optimally allocate those flows.

6.5 MAP Mobility

When looking at the mobility aspects of the MAP implementation, the protocol itself does not provide any mobility management; however, each host in the mobile network is provided with an improved view of the available connectivity. Therefore, MAP puts the impetus on the host to take control of the additional information, taking advantage of the additional resource advertised. Throughout this thesis we have focused on MPTCP to provide the host with both resource pooling, in this section we will demonstrate the benefits to host mobility that can be gained from augmenting MPTCP with MAP. To this end, we will analyse and evaluate the improved flexibility that MAP can provide in regards to MPTCP mobility.

In this experiment we reuse the previous real world topology shown in Fig. 6.20 on page 184, however we extend the topology by introducing an additional user device (**U6**), with additional network resource shaped according to L1 in Table 6.7 on page 183. MAPD is configured to transmit heartbeats every second (60 HBPM), while link timeouts are scheduled every two seconds. Each host creates two long-lived TCP connections using iPerf, attempting to maximize utilisation of the available bandwidth. In Table 6.9, we present the set of events that occur during this mobility scenario. At T0, each of the user devices from

U1 to U5 initiate an iPerf connection to their own remote server, at this point in time, the Ad-Hoc mesh supporting communications between H1, H2, and H3 is not connected. At time T40, H1 and H3 are able to connect establishing the first network link in the mesh. At T70, H1, H2, and H3 are all able to communicate via the mesh network. Subsequently at time T90, U6 associates with H1's access point, including the network resource L1 into the mobile network, before establishing its own iPerf connection. At T130, connectivity to L4 is lost by H3. T150 sees U4 disassociate from H2, performing a hard handover, subsequently associating with H1. Finally H3 establishes a network connection via L7 at T190 adding additional resource to the mobile network. Finally at T210, the experiment finishes.

6.5.1 Results

The outcome of the mobility scenario is presented in Fig. 6.22 on the next page. The mobility events and the resulting MAP advertisements demonstrate the additional flexibility that is introduced through the dissemination of network resource. In the case of U3 and U5 at T40, both devices are informed and subsequently establish MPTCP subflows across both L4 and L5, within two seconds of the mesh link being brought up. This is mirrored by U1 creating a subflow for L8 which is also announced on the mesh link; however, the throughput for U1 over this subflow remains low. The same behaviour is seen for U2 and U4 at T70, both users quickly react to the update creating the appropriate additional subflows. U1, U3 and U5 are slower to initialise an additional subflow after H2's announcement of L10, which is likely caused by the poor quality of the link and an increase in congestion delaying the handshake. When U6 associates with H1 at T90, all other user devices are quick to establish new subflows, with a sub second delay for U1 to U5 to create subflows over L1. At time T130, H1 loses connectivity to L4, while this action would force all flows using this network resource to stop, the removal via MAP will force the flows to be closed faster than simply waiting for them to time out. When U4 performs the hard handover from H3 to H1 at T150, new subflows are re-created quickly using the new IP addresses provided by MAP, the association with H1 itself takes an average of 5 seconds, from the disconnection being triggered. After association, U4 first performs a DHCP request to obtain an IP address for internal communications, which must occur before the MAP

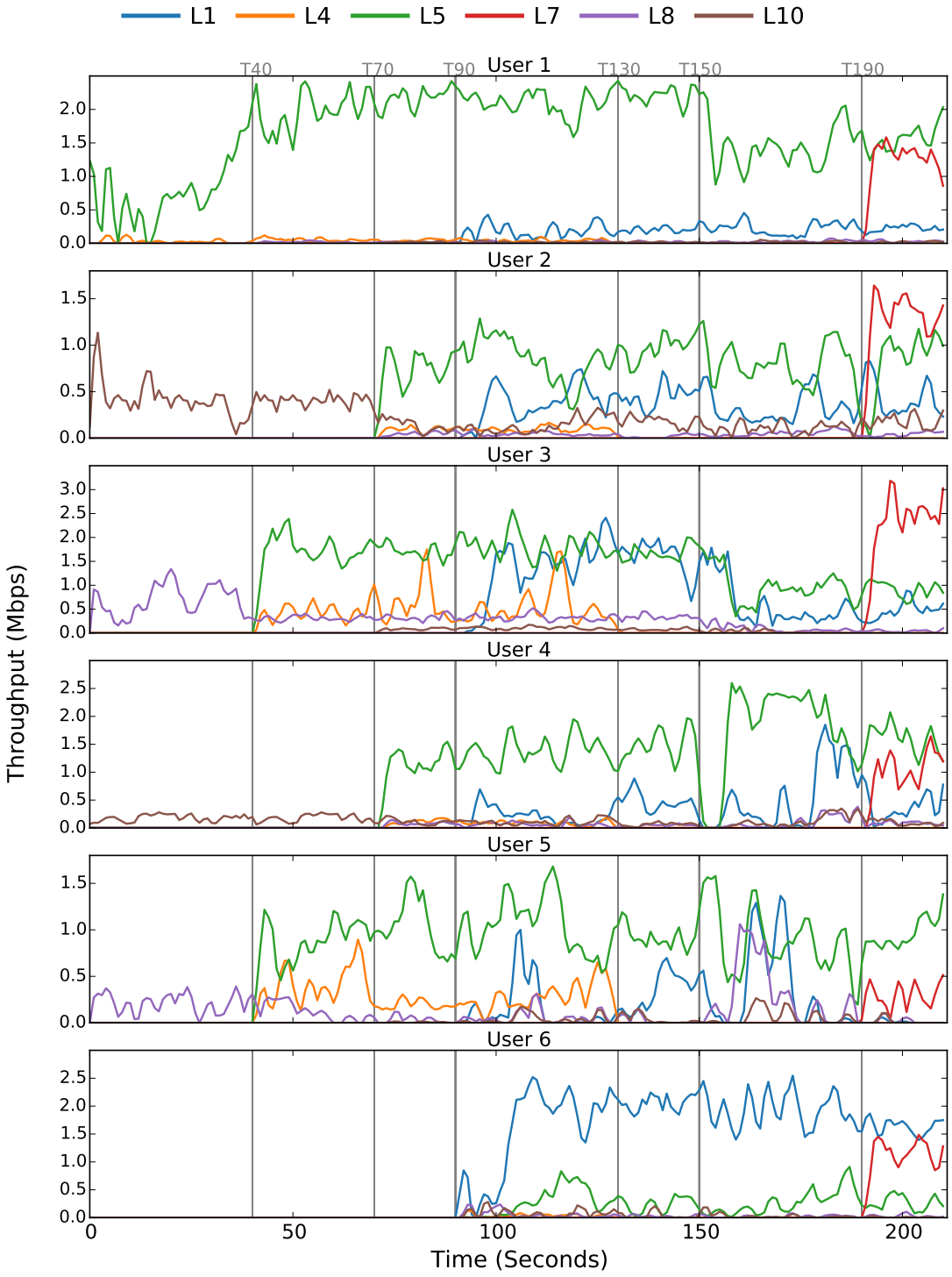


Figure 6.22: Network resource usage during the mobility scenario for each individual host.

request is transmitted. In this case, MAP adds a small sub second delay before the new TCP connections are created. Finally at T190, L7 is advertised by H3; due to the lower delay and higher bandwidth, it is quickly incorporated into the users MPTCP connection, increasing total throughput for each user.

It is interesting to note that the hosts in the mobile scenario, typically prefer the connectivity that is closer to them, as demonstrated by U6 which utilizes L1 much more than the faster L5. This is not always the case, and interesting behaviours have emerged; for example, both U3 and U5 utilize L4 much more than U1, despite it being an additional hop away. Furthermore, in this experiment, H5 presented the most inconsistent throughput; which is likely to be caused by the increased number of hops and subsequently increased internal congestion.

This experiment has shown that the benefits of MAP are not only limited to increasing network throughput, but through advertisement of additional network resource, hosts and their associated flows become more flexible. This additional flexibility allows hosts to adapt more quickly to changes in the network; for example, a gateway may change its point of attachment to the Internet. Typically, with both TCP and MPTCP this would result in the connection breaking. With MAP however, if a host or gateway in the mobile network changes its point of attachment, the new resource is announced and subsequently triggers a new MPTCP subflow to be created leading to a soft handover of indirect network resource. In the context of a mobile network, this is incredibly valuable functionality as hosts may change their point of attachment to the Internet regularly.

6.6 User Policy Framework

In this section we will present the benefits that can be drawn from both the context awareness and the resource management components of the User Policy Framework implementation. This functionality allows the user to describe exactly how and when their available network resource should be used. The use of context enables novel management approaches to be used such as preempting handovers. We demonstrate that the User Policy Framework provides an enhanced approach to resource management, improving on the simple schemes that are typically seen today.

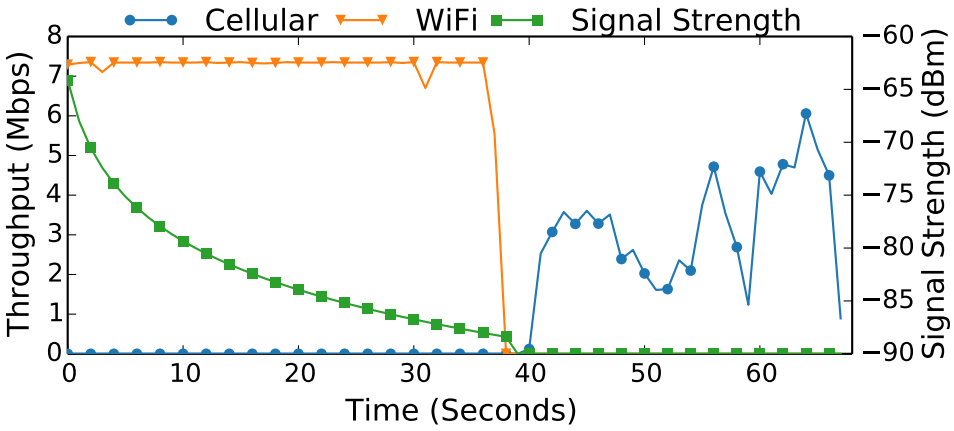
6.6.1 Preempting Disconnections

In Fig. 6.23 on the next page, a context module has been implemented to monitor the signal strength associated with a WLAN connection. The context configuration file for this event is presented in Listing 6.1 on page 192. This example relies on the extended MPTCP user space path manager presented in Chapter 5, allowing the User Policy Framework to dictate establishment of subflows.

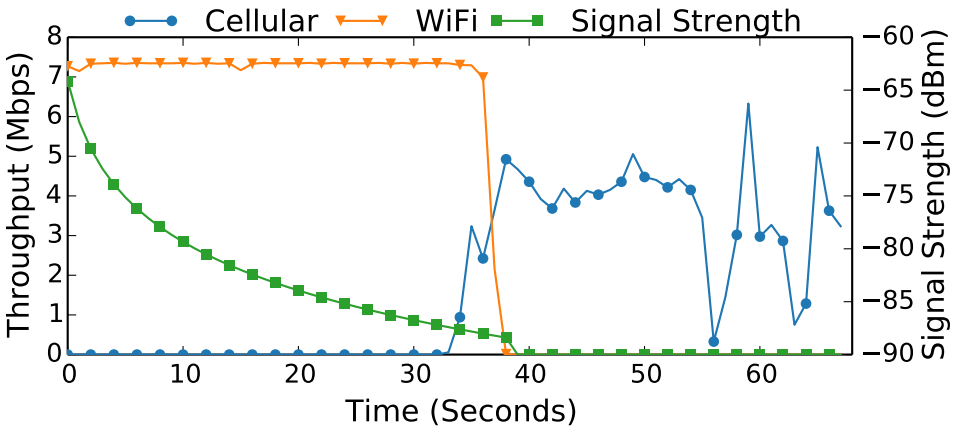
In the results shown in Fig. 6.23 on the next page, the cellular interface is set to act as a backup to be used when the WiFi interface disconnects. To determine disconnection, we emulate signal strength using the model proposed in [40], with the user moving at 1.4 Meters per second away from the WiFi access point. When the signal strength reaches approximately -88dBm, the WiFi interface is disabled. In Fig. 6.23a, the behaviour of the stock MPTCP implementation is shown. When the WiFi interface disconnects it takes up to two seconds for the cellular subflow to start transmitting data, leading to the throughput dropping to zero for these two seconds, corroborating the results presented in [122]. In Fig. 6.23b, the signal strength module is monitoring the connectivity information for wlan0, as the quality drops, approaching the point of disconnection, a new subflow is preemptively created over the cellular interface. As the cellular subflow is created before the WiFi disconnects the throughput does not drop to zero, improving the seamless nature of the handover and therefore the users experience.

6.6.2 Adapting to battery capacity

Mobile devices are limited in terms of battery capacity, therefore it is important to not only maximize battery life, but to also effectively balance this while trying to maximize the users quality of experience. Listing 6.2 on page 194 presents the User Policy Framework configuration to adapt resource allocation based on battery capacity. In the results shown in Fig. 6.24 on page 193, the user is assumed to be watching a movie using Adaptive HTTP Streaming while mobile. As the amount of bandwidth changes the quality of video that the user receives adapts to minimize re-buffering. Both the WiFi and cellular interfaces have been enabled to maximize the quality of the video stream that is being viewed. This increases the total energy consumption required to view the video, and depending on the length, the battery capacity could be run down prematurely. When using



(a)

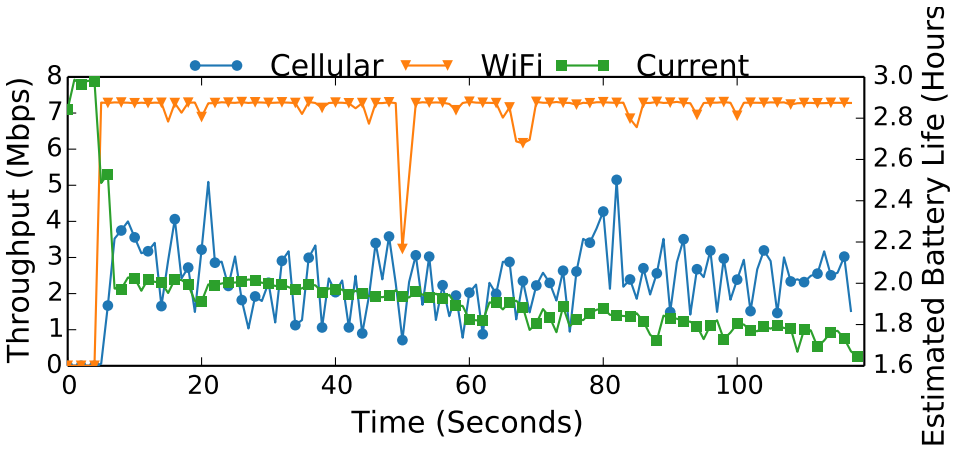


(b)

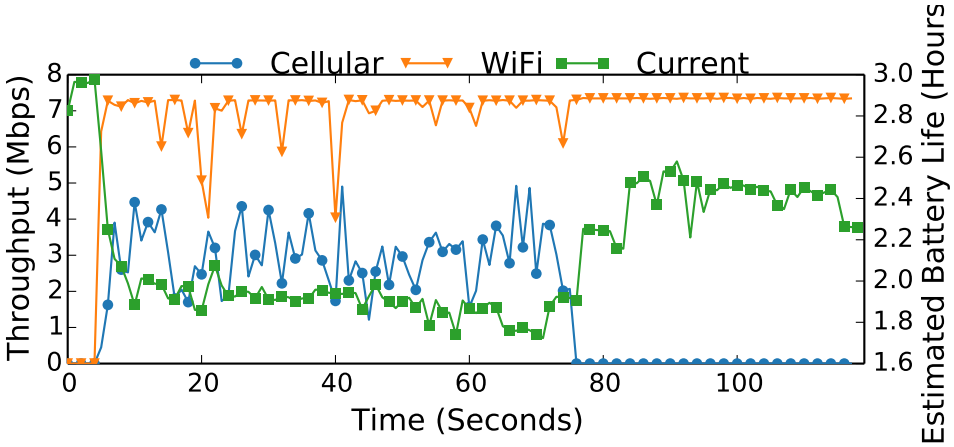
Figure 6.23: Handover of WiFi to Cellular with (a) MPTCP and (b) Context Driven MPTCP

```
[
    {
        "policy": {
            "condition":[
                {
                    "link_id": "wlan0",
                    "key_id": "wlan_rssi",
                    "value":"-88.00",
                    "comparator":"<="
                }
            ],
            "action":[
                {
                    "do":"enable",
                    "link_id":"ppp0",
                    "mode":"hard"
                }
            ]
        }
    }
]
```

Listing 6.1: Policy file configuration to preemptively migrate traffic to the cellular network interface before the WLAN interface disconnects.



(a)



(b)

Figure 6.24: Energy consumption while streaming video for (a) MPTCP and (b) Context Driven MPTCP

```
[
    {
        "policy": {
            "condition":[
                {
                    "link_id": "any",
                    "key_id": "battery_capacity",
                    "value":"50%",
                    "comparator":"<="
                }
            ],
            "action":[
                {
                    "do":"disable",
                    "link_id":"ppp0",
                    "mode":"hard"
                }
            ]
        }
    }
]
```

Listing 6.2: Policy file configuration to migrate traffic away from the ceullar network interface when power consumption drops below 50%.

the full mesh path manager as shown in Fig. 6.24a, both interfaces are used indefinitely for the duration of the video. In this state, the total current draw for the Raspberry Pi, including WiFi and Cellular is approximately 0.96 Amps. At this rate the device would only remain turned on for a little over 2 hours, given the 2000mAh battery used. In Fig. 6.24b, the User Policy Frameworks context modules identify that the battery capacity is too low and the current draw is too high to maintain, this subsequently triggers an event causing the Cellular subflow to be closed, allowing the network interface to fall into a sleep state, dropping the average current draw from 0.96 Amps to 0.73 Amps, extending the lifespan of the battery, from approximately 1.6 hours to 2.2 hours by the end of the experiment. Due to the adaptive application context, reducing the available bandwidth leads to a lower video quality as opposed to significant levels buffering or broken connections. Decreasing video quality in order to increase battery life may be beneficial to the user, especially if it enables the device to stay powered until the next charge is available.

6.6.3 Migrating traffic based on priority

The core concept of this context is based on traffic priority, a user device may decide that a specific application, or the application they are currently interacting with should receive a higher proportion of the available bandwidth, in comparison to the background applications. The User Policy Framework configuration for this context is presented in Listing 6.3 on the next page.

In the results shown in Fig. 6.25 on page 197, an initial application (App One) is in the background actively downloading content, this could represent receiving software updates or syncing files with cloud storage. After 20 seconds the user starts a second application (App Two), App Two then proceeds to download a small 10MB file that the user is interested in, so should be downloaded as quickly as possible. In Fig. 6.25a on page 197, both App One and App Two establish MPTCP connections, creating subflows for both the WiFi and cellular network interfaces. When App Two becomes active, both applications share the available bandwidth equally, leading to App Two taking 18 seconds to download its content. In Fig. 6.25b on page 197, a policy has been defined that states App Two should receive priority over App One in terms of bandwidth. When App One starts, the User Policy Framework is aware that both network interfaces are free to be used,

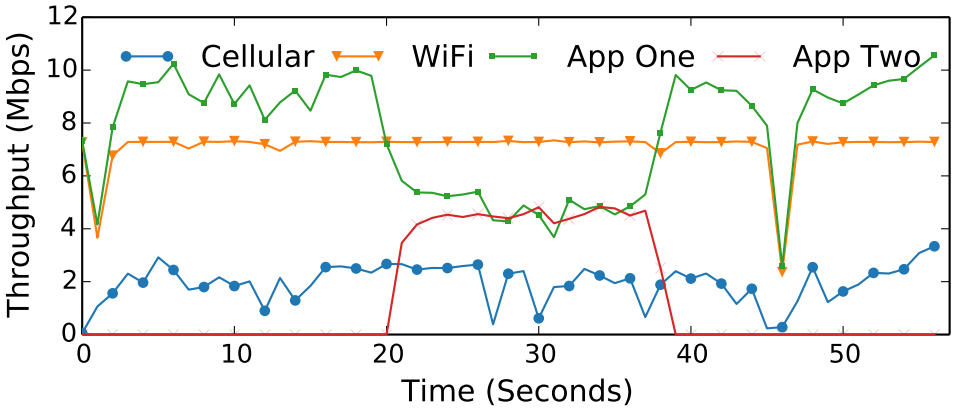
```
[
    {
        "policy": {
            "condition":[
                {
                    "link_id": "any",
                    "key_id": "prioritize",
                    "value":"current_application",
                    "comparator":"=="
                }
            ],
            "action":[
                {
                    "do":"handover",
                    "link_id":"ppp0",
                    "mode":"soft"
                }
            ]
        }
    }
]
```

Listing 6.3: Policy file configuration to prioritise the users current traffic over background traffic.

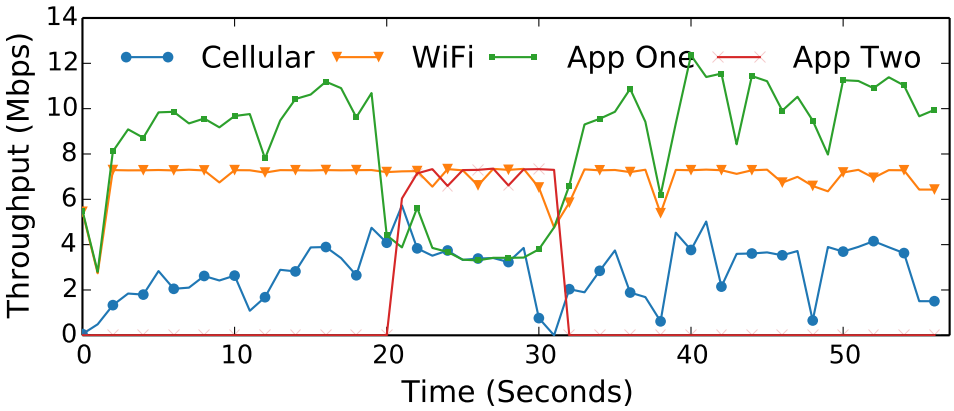
and subsequently creates subflows for each path. When App Two starts, the User Policy Framework gives App Two priority by removing App One's WiFi subflow. This forces App One to only use the cellular interface, giving App Two exclusive access to the higher bandwidth WiFi interface. This reduces the time taken to download the users file to 11 seconds, an improvement of 38%.

6.6.4 Collaborative Policy Results

We now present a combination of policies that are applied to the devices in the mobile network in Fig. 6.20 on page 184. The first contribution of this evaluation is to show how sophisticated resource usage policies can be used to better meet the needs of the user, by quickly adapting to the state of the device. The second contribution illustrates how MAP can be leveraged alongside the policy framework as the user devices react to changes in network resource based on the dissemination approach. To this end we present three policies. H1's first policy



(a)



(b)

Figure 6.25: TCP flows competing for network resource for (a) MPTCP and (b) Context Driven MPTCP. App One represents background activity and App Two represents the users active application.

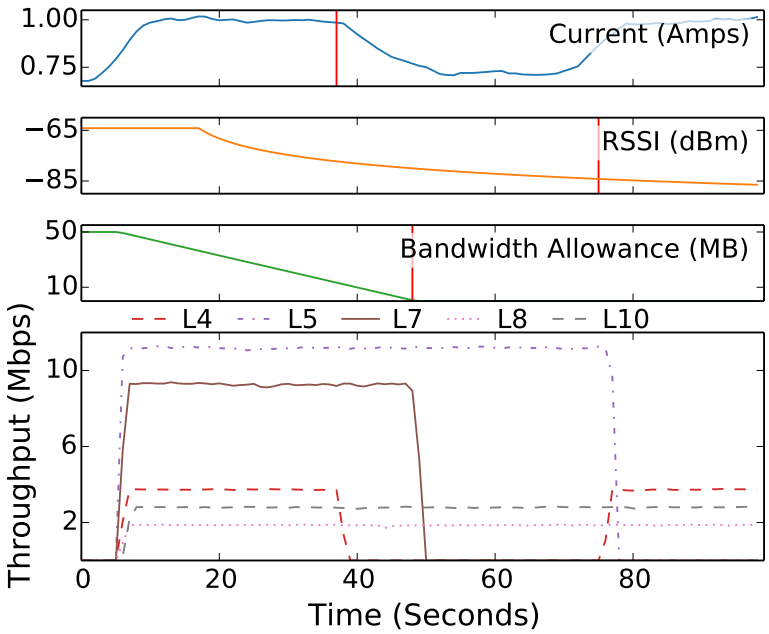


Figure 6.26: Context modules triggering policy decisions in a cooperative mobile environment.

($P1$) states that battery life should be extended, therefore when the capacity drops to 50%, traffic should be moved away from the cellular interface if an alternative is available. H1's second policy ($P2$) states that when the signal strength of the WiFi connection degrades to -89dBm the backup connection should be brought up. Finally H3 specifies an allowance policy, ($P3$), only wanting to send 50MB of data over the WiFi interface if the SSID is equal to *DEMO_AP_TWO*. To show the impact of these policies each user device loads the network with a single TCP flow, using iPerf. To ensure repeatability, the context modules read emulated values for the battery capacity and signal strength. The results of this experiment are shown in Fig. 6.26, with the values of the contexts being illustrated in the upper three plots and the trigger activation is represented by the red vertical line. When the policy framework identifies that the link is no longer available to be used, based on policies $P1$ and $P3$ (low battery and allowance used), the update is disseminated to all hosts in the network. $P3$ then indicates that the cellular link can be used again, as handover is preempted, the policy framework at each host initiates a call to the MPTCP kernel module, indicating that a new subflow should be created. In this evaluation we have shown how ex-

ploiting context to create descriptive resource management policies can improve the potential for a mobile device to meet the users needs; enabling power saving, resource pooling and pre-emptive handover on demand.

6.7 Path Selection

In this final evaluation, we present the quality of service and experience benefits that can be drawn from the proposed path selection algorithm. Continuing with the same real world testbed shown in Fig. 6.20 on page 184, making all connectivity available using MAP. Furthermore, each of the users in the local network runs an instance of the User Policy Framework implementation with the proposed PCA path selection algorithm enabled. Initially we begin the path selection evaluation with a static analysis of the algorithm, subsequently we observe a real time experiment with cross traffic and multiple applications.

During the static analysis of the path selection algorithm, the primary goal is to compare the quality of service and quality of experience that an application receives over each of the available network links. Then given this comparison, demonstrate that for each of the applications and links, the PCA algorithm has made an optimal selection improving the QoE or QoS the user receives.

In keeping with the evaluation thus far, the analysis of path selection and the QoS and QoE results are derived using the same real network topology shown in Fig. 6.20 on page 184. User Three specifies an application policy for a live RTP video stream (***A1***) according to Table 6.10 on the next page. The RTP video transmission is built on gstreamer [62], streaming the Big Buck Bunny stock video in 480P. We run the policy framework with the PCA selection algorithm, which selects ***L5*** for ***A1***. This selection was tested and proven by calculating the self similarity (SSIM) [152] of the video transmitted over all available paths, which is shown in Table 6.11 on page 201, alongside the PCA utility score. In this case, the link that provides the highest quality video stream according to the SSIM measurement is L5, with a value of 0.704. With the presented context, this demonstrates the optimal path selection decision. Subsequently, the second best path in terms of QoE also represents the second choice from the PCA algorithm. This pattern of successful ranking continues for the third best link, however the fourth and fifth are switched in regards to the QoE against the path selection

	Type	**BW (Mbps)**	**RTT(Ms)**	**Loss (%)**	**Jitter (Ms)**
A1	Video	8.000	50.000	0.010	30.000
A2	Sensor	0.010	300.000	0.00	300.000
A3	Unspec	-	-	-	-

Table 6.10: Application specifications and the links chosen from Table 6.7 on page 183 for communication, according to the PCA algorithm.

ranking. This demonstrates that the PCA approach may not always provide optimal and flawless results.

For the second experiment, User Two specifies a real time sensor application (***A2***) according to Table 6.10. For this application the inter-arrival time of packets should be low, with each of the packets arriving within a second of the last, all links are deemed to be suitable by the metric estimation, so a lower quality path can be used. To measure quality of service we simply look at the percentage of packets that arrive on time (AT). The application used as the model for A3, is a bespoke lightweight UDP client, that transmits a small JSON packet containing current GPS coordinates, a unique device identifier, and the current time stamp. To determine the quality of service, in terms of packet arrival, we transmit 1000 packets.

In the final experiment, User One defines the need for the best overall path for application (***A3***) i.e. the application requirements are left empty; however, there is significant cross traffic from another application using ***L5***, introduced by iPerf, reducing the perceived quality. The PCA algorithm subsequently allocates application ***A3*** to ***L7***. The quality of service received is defined in terms of the bandwidth that application ***A3*** receives. As ***L7*** is limited by the introduction of cross traffic, the estimated quality of each of the network metrics is reduced. This helps to demonstrate that obtaining accurate and up to date network measurements can help to determine the most appropriate network resource, ensuring the success of the PCA path selection algorithm.

6.8 Summary

In this chapter, we have presented a range of evaluation and experimentation in the mobile network domain. Initially we focused on the scalability implications of the Linux MPTCP implementation. The scalability experiment demonstrated

		Network Resources							
App	**Value**	**L4**	**L5**	**L7**	**L8**	**L10**	**Default**	**Selected**	**Best**
A1	SSIM	0.384	0.704	0.610	0.334	0.378	L8	L5	L5
	US	1.033	1.584	1.506	0.754	0.00			
A2	AT	100	100	100	100	100	L8	L10	Any
	US	0.565	0.255	0.026	0.816	1.384			
A3	Mbps	2.683	5.73	7.267	1.828	2.080	L5	L7	L7
	US	0.977	1,488	1.564	0.695	0.00			

Table 6.11: Quality of service and experience ratings for a set of applications and the calculated PCA utility scores (US), with the default and selected links.

that MPTCP is still limited in comparison to single-path TCP, with coupled congestion control suffering from decreased throughput in comparison to CUBIC. Despite this, the flexibility and adaptability that can be achieved through MPTCP in comparison to single-path TCP is significant; in many cases, single-path TCP will not be able to utilise the same number of paths as the multipath variant, without explicit intervention by the user or through modifying applications, negating the gap in performance. Subsequently, we presented the behavioural aspects of MAPD, specifically looking at packet overhead, convergence time, power requirements, and processing time. While improvements could be made to the performance aspects, as a prototype the negative impact appears negligible, and the convergence time is minimal in comparison to the benefit gained from access to the additional network resource.

We then proceeded to analyse the performance benefits of MAPD in comparison to other connectivity approaches, in small to medium size mobile networks, from two to sixteen hosts. This demonstrated that MAPD with MPTCP provides sufficient gains in throughput in the proposed topologies, the more free network resource that is available the more a host can gain from using MAPD. Despite this, with small sizes of TCP connection, the overhead of using MPTCP over 8 to 16 links, outweighed the benefit of increased capacity. We then presented the same problem in the real world, analysing the behaviour for a mobile network based on the fire service use case presented in Chapter 1. The results of the real world network corroborated our findings in the simulation environment, demonstrating that MAPD with MPTCP becomes more useful as the size of the TCP connection increases. We then proceeded to analyse the mobile network under mobility conditions, in which hosts connect and reconnect to different ac-

cess points, and link state changes. We compared MAP and MPTCP to a tunnel based network mobility model, this demonstrates that MAP provides more flexibility, however it adds additional delay to a handover while the MAPD request and update process occurs.

Finally, the evaluation perspective moved on from MAP, to observe how network resource usage could be improved from the perspective of a single host, evaluating the implemented User Policy Framework and path selection algorithm. To this end, we demonstrated a variety of contexts that are supported by the extensible framework, which can help a user dictate and control how their network resource is used, helping their device to meet their specific needs. The evaluation concluded demonstrating the ability for our proposed PCA based path selection algorithm to allocate applications described in terms of network requirements to a specific link, in all cases, the PCA algorithm predicted the best possible route based on network metric estimations.

CHAPTER 7

Conclusion

This thesis has presented the case for cooperative resource pooling in mobile networks. The state of the art in resource pooling was surveyed presenting a wide range of tools, techniques, and protocols that can be used to support a cooperative approach. Furthermore, we presented the current state of the art in terms of collaboration and cooperation approaches for both users and mobile networks, which fundamentally did not address the problem domain, that can be solved using resource pooling. We established a network resource sharing protocol, with a focus on facilitating cooperation, which combines with the MPTCP resource pooling approach. This concept led to the design and implementation of the Multipath Advertisement Protocol. The protocol implementation aims to leverage multiplicity and diversity available in a multihomed mobile network, bridging the gap between the single user resource pooling approaches and collaborative Internet access that is currently of interest within the research community. Furthermore, we augmented the MAP protocol with a policy framework and path selection algorithm, in an effort to improve the utilisation of the available network resource.

7.1 Summary

Looking back at the initial aims of this thesis, there were three core contributions which included: a routing overlay to enable resource sharing in a mobile network; an extensible policy framework to improve management of network resource; and a path selection algorithm to improve the quality of experience. For the remainder of this section, we present an overview from design to evaluation, of each of the

three core contributions:

1. **Multipath Advertisement Protocol** - The underlying focus for this thesis has been to support the sharing of network resource between mobile hosts. To facilitate this sharing, we designed and implemented a prototype of the Multipath Advertisement Protocol (MAP). The implementation of MAP (MAPD), provides each host with additional addresses allowing them to communicate via overlay routes on top of the existing network. Therefore MAPD is able to seamlessly support hybrid networks, bridging ad-hoc and infrastructure, which may be using a diverse set of underlying routing protocols such as AODV or OLSR. Given this implementation, we have evaluated the overhead of MAPD, showing that the overhead scales linearly, with the number of hosts, and to a lesser extent the number of network resources in the network. Furthermore, we have demonstrated how MAPD can be combined with MPTCP to enable cooperative resource pooling in a mobile network. Initially we focused on evaluating the benefits of MPTCP at scale, using different congestion control algorithms, in comparison to single path TCP. The scalability of MPTCP, showed that there are still improvements that can be made, to both scheduling and congestion control components, as single path TCP consistently outperformed MPTCP, without competing subflows. The following evaluation measured the throughput achieved in a mobile network using a collection of different resource pooling approaches. As the size of the TCP flows increased, the benefits of cooperative resource pooling increased.

2. **User Policy Framework** - Typically hosts are not able to optimize or appropriately manage the available network resource. Smart devices capture a wealth of information, that can be used to better establish a model for users connectivity, based on the users and the devices context. The policy framework provides an extensible approach to the management and control of network resource for a Linux based host. The policy framework supports the dynamic loading of context modules. These modules are able to monitor and process specific contexts regarding the user, host, or network. The ability to easily model, monitor, and subsequently respond to specific contexts will provide an effective grounding for future research, facilitating improved network management schemes. To provide improved management

of network resource, a bespoke MPTCP path manager was designed and implemented that interfaces with the policy framework to determine how subflows should be created. Using the policy framework, we presented a proof of concept evaluation, showing the value of an extensible framework using a subset of demonstrable contexts.

3. **Path Selection Algorithm** - As each of the hosts in a cooperative mobile network are able to access additional network resource, the ability to select the optimal path (or set of paths) for any specific application can improve the quality of experience for the user. The path selection algorithm is based on the concept of Principal Component Analysis (PCA), which is able to reduce the dimensionality of the metrics an application may be interested in. Additionally, PCA is able to remove the correlation between variables. Disregarding the correlation between variables is of significant importance, as it may distort and provide skewed results when determining the most appropriate path for pre-specified application. We presented a proof of concept evaluation, demonstrating that path selection can improve the quality of experience or quality of service with respect to choosing the default link in the network. Using the PCA based approach provides a more sophisticated approach to path selection in comparison to more recent work in the mobile network domain as presented in [4].

7.2 Future Work

In this section we reflect on the output and completeness of the presented work; furthermore, as the concept of cooperative resource pooling, in the proposed context is still in its infancy, we detail the future areas that will be of importance when continuing to implement and improve on the presented cooperative resource pooling approach. We will consider future work, looking at individual components of the three main contributions: MAP, the User Policy Framework, and path selection.

7.2.1 IPv6

The proposed design and implementation of MAP has focused completely on an IPv4 testbed, the initial reasoning for this was testing and observing the performance of the cooperative resource pooling approach using real world networks, which have limited IPv6 support, especially in the mobile domain. While all of the proposed techniques can be directly translated from IPv4 to IPv6, the differences between the two protocols could lead to a more appropriate model existing for IPv6. Our prototype implementation relies on using NAT at each hop to ensure the routability of each of the additional network resources. The use of NAT in an IPv6 context however is controversial, as it breaks the goal of pure end to end connectivity. Making each host globally routable, via each of the network interfaces in a multihomed mobile network could have a significant and negative impact, and could require standardisation of subnet provision to reduce the potential overhead. The alternative would be to take a tag based approach [16], marking packets to be routed, relying on a bespoke MPTCP path manager to manage the creation and deletion of MPTCP subflows, as opposed to being able to transparently use an appropriate MPTCP path manager.

7.2.2 Coupled Congestion Control

One of the key observations made during the evaluation of MPTCP and MAP showed the current inadequacies of using coupled congestion control in the proposed scenario. Primarily without any competing traffic in the first experiment, and more specifically without a shared bottleneck, MPTCP with CUBIC and either OLIA or LIA both consistently underperformed in comparison to single path TCP with CUBIC. As specified in [135], the goal of MPTCP is to obtain at least the bandwidth of singlepath TCP over the best path. In the absence of competing TCP flows this goal has been met, however it is also very conservative, improving utilisation further should be possible increasing the benefits of a multipath transport layer. When comparing CUBIC to OLIA across each of the hosts, we anticipated CUBIC to consistently perform better, providing a higher throughput, with OLIA demonstrating the power of coupled congestion control by decreasing the range of throughputs achieved across the set of hosts. The benefits of shared bottleneck detection however, was not consistently proved

and the hosts with access to more network resource typically obtained more than their fair share of bandwidth, limiting the hosts with access to less. If the aim is to allows multiple hosts to share network resource, this could open an avenue of research to further improve congestion control algorithms for such a scenario.

7.2.3 Policy Definition

While the current policy definition is simple and descriptive enough to support a wide range of user and device policies, managing conflict and policy prioritisation has effectively been left as an exercise to the user. Building a system that is able to intelligently prevent and resolve policy conflicts could be of significant interest to the research community, with external applicability to a number of areas. The first step to improving policy management in this regard is to introduce a prioritisation system, such that, if two policies conflict simultaneously, the higher priority policy will always succeed. Furthermore, an advanced policy definition should additionally have the ability to detect and prevent circular dependencies of policies, in which a set of policies may continuously trigger one another, preventing the system from stabilizing.

7.2.4 Path Selection

The design and implementation of the path selection approach, demonstrates that selecting paths for applications can improve the quality of service, or quality of experience received. The pluggable approach to path selection algorithms presented in the policy framework provides a good grounding for future research in this area, allowing researchers to use the underlying implementation to easily evaluate alternative approaches. Our evaluation while demonstrating the benefits of our approach, did not provide results comparing to other path selection approaches such as SAW as presented in [4]. Furthermore, path selection could be used to better inform more choices about resource usage, instead of simply allocating applications to a single path; for example, the path selection could infer the best subset of paths to use for an MPTCP connection, or identify which resources are redundant in the current context. Finally extending the path selection approach to include additional metrics such as availability, reliability, and cost could further improve the users quality of experience beyond looking at the direct

quality of the network. Such an evaluation could help to cement the concept of path selection for multihomed network resource.

7.3 Final Words

The overarching aim of this thesis has been to prove and demonstrate that an enhanced approach to allocating and managing network resource in a cooperative environment can improve the quality of service for users in the network. This problem domain has become more important as the demand for high bandwidth applications, including serving media and providing cloud storage, has continuously increased. The core infrastructure facilitating this demand is continually improving, while at the edge, users typically rely on hardware upgrades to move to the next generation technology, which may leave a gap between supply and demand. To this end, it is becoming evermore important to make better use of the network resource that is available, through cooperation, resource pooling, and improved path management and selection.

Bibliography

[1] L. Abdullah and C. R. Adawiyah. Simple Additive Weighting Methods of Multi criteria Decision Making and Applications: A Decade Review. *International Journal of Information Processing and Management*, 5(1):39–49, 2014.

[2] H. Adhari, T. Dreibholz, M. Becke, E. Rathgeb, and M. Tuxen. Evaluation of Concurrent Multipath Transfer over Dissimilar Paths. In *IEEE Workshops of International Conference on Advanced Information Networking and Applications*, WAINA, pages 708–714, March 2011.

[3] B. Albrightson, J. J. Garcia-Luna-Aceves, and J. Boyle. EIRGP - A Fast Routing Protocol Based On Distance Vectors. In *Proceedings of Networld/Interop*, 1994.

[4] I. Alsukayti and C. Edwards. Multihomed Mobile Network Architecture. In *IFIP Networking Conference*, pages 195–203. IFIP, 2015.

[5] G. Ananthanarayanan, V. Padmanabhan, C. Thekkath, and L. Ravindranath. Collaborative Downloading for Multi-homed Wireless Devices. In *Proceedings of the 8th IEEE Workshop on Mobile Computing Systems and Applications*, HotMobile, pages 79–84, March 2007.

[6] Apple. iOS: Multipath TCP Support in iOS 7. `https://support.apple.com/en-us/HT201373`. [Online; Accessed: 2015-07-31].

[7] U. Ashraf, S. Abdellatif, and G. Juanole. Gateway Selection in Backbone Wireless Mesh Networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, WCNC, pages 1–6, April 2009.

[8] R. Atkinson and S. Bhatti. Identifier-Locator Network Protocol (ILNP) Architectural Description. RFC 6740 (Experimental), November 2012.

[9] B4RN. Broadband For the Rural North (B4RN). http://b4rn.org.uk/. [Online; Accessed: 2015-07-31].

[10] A. Baird and N. Wright. Poor Access to Care: Rural Health Deprivation? volume 56, pages 567–568. British Journal of General Practice, 2006.

[11] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting Mobile 3G Using WiFi. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys, pages 209–222, New York, NY, USA, 2010. ACM.

[12] R. Bellman. A Markovian Decision Process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

[13] M. Benoliel, S. Shalunov, and G. Hazel. Open Garden. `www.opengarden.net`. [Online; Accessed: 2015-07-31].

[14] M. Benzaid, P. Minet, K. Agha, C. Adjih, and G. Allard. Integration of Mobile-IP and OLSR for a Universal Mobility. *Wireless Networks*, 10(4):377–388, Jul 2004.

[15] G. Biczók, L. Toka, A. Vidacs, and T. Trinh. On Incentives in Global Wireless Communities. In *Proceedings of the 1st ACM Workshop on User-provided Networking: Challenges and Opportunities*, U-NET, pages 1–6, New York, NY, USA, 2009. ACM.

[16] J. Boite, V. Conan, G. Nguengang, A. Ploix, and D. Gaiti. Lightweight Tag-Based Forwarding Among Competing Gateways in Wireless Mesh Networks. In *IEEE Wireless Communications and Networking Conference*, WCNC, pages 2157–2162, April 2012.

[17] C. Boldrini, M. Conti, F. Delmastro, and A. Passarella. Context- and social-aware middleware for opportunistic networks. *Journal of Network and Computer Applications*, 33(5):525 – 541, 2010.

[18] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. *SIGCOMM Comput. Commun. Rev.*, 24(4):24–35, Oct 1994.

[19] J. Broch, D. Maltz, and D. Johnson. Supporting Hierarchy and Heterogeneous Interfaces in Multi-Hop Wireless Ad Hoc Networks. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks*, ISPAN, pages 370–, Washington, DC, USA, 1999. IEEE Computer Society.

[20] L. Buttyán and J.-P. Hubaux. Stimulating Cooperation in Self-organizing Mobile Ad Hoc Networks. *Mob. Netw. Appl.*, 8(5):579–592, Oct 2003.

[21] Y. Cao, M. Xu, and X. Fu. Delay-based congestion control for multipath TCP. In *20th IEEE International Conference on Network Protocols*, ICNP, pages 1–10, Oct 2012.

[22] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Pocket Switched Networks: Real-world mobility and its consequences for opportunistic forwarding. Technical Report UCAM-CL-TR-617, University of Cambridge Computer Laboratory, August 2005.

[23] Y. Chen, T. Farley, and N. Ye. QoS Requirements of Network Applications on the Internet. *Inf. Knowl. Syst. Manag.*, 4(1):55–76, Jan 2004.

[24] Y. Chen, E. Nahum, R. Gibbens, D. Towsley, and Y. Lim. Characterizing 4G and 3G Networks: Supporting Mobility with Multi-Path TCP. Technical report, University of Massachusetts Amherst, University of Massachusetts, Amherst, Massachusetts, U.S.A, Sep 2012. [Online; Accessed: 2015-07-31.

[25] J. Chroboczek. The Babel Routing Protocol. RFC 6126 (Experimental), April 2011. Updated by RFCs 7298, 7557.

[26] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis. Increasing TCP's Initial Window. RFC 6928 (Experimental), April 2013.

[27] Y. Chuang and K.-J. Lin. Cellular Traffic Offloading Through Community-based Opportunistic Dissemination. In *IEEE Wireless Communications and Networking Conference*, WCNC, pages 3188–3193, April 2012.

[28] S. Chupisanyarote, S. Kouyoumdjieva, O. Helgason, and G. Karlsson. Caching in opportunistic networks with churn. In *9th Annual Conference on Wireless On-demand Network Systems and Services*, WONS, pages 39–42, Jan 2012.

[29] Cisco. Cisco Visual Networking Index: Forecast and Methodology, 20142019, May 2015. Accessed: 2015-07-31.

[30] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 20142019, February 2015. Accessed: 2015-07-31.

[31] T. Clausen, C. Dearlove, and J. Dean. Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP). RFC 6130 (Proposed Standard), April 2011. Updated by RFCs 7183, 7188, 7466.

[32] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg. The Optimized Link State Routing Protocol Version 2. RFC 7181 (Proposed Standard), April 2014. Updated by RFCs 7183, 7187, 7188, 7466.

[33] Connectify. Speedify. `http://speedify.com/`. [Online; Accessed: 2015-07-31].

[34] M. Conti and M. Kumar. Opportunities in Opportunistic Computing. *Computer*, 43(1):42–50, Jan 2010.

[35] M. Coudron, S. Secci, G. Pujolle, P. Raad, and P. Gallard. Cross-layer cooperation to boost multipath TCP performance in cloud networks. In *IEEE 2nd International Conference on Cloud Networking*, CloudNet, pages 58–66, Nov 2013.

[36] J. Crowcroft, R. Gibbens, F. Kelly, and S. Östring. Modelling Incentives for Collaboration in Mobile Ad Hoc Networks. *Perform. Eval.*, 57(4):427–439, Aug 2004.

[37] T. Davis, W. Tarreau, C. Gavrilov, C. Tindel, J. Girouard, and J. Vosburgh. Linux Bonding. `http://www.linuxfoundation.org/collaborate/workgroups/networking/bonding`. [Online; Accessed: 2015-07-31].

[38] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris. A High-throughput Path Metric for Multi-hop Wireless Routing. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, MobiCom, pages 134–146, New York, NY, USA, 2003. ACM.

[39] N. Do, C. Hsu, and N. Venkatasubramanian. CrowdMAC: A Crowdsourcing System for Mobile Access. In *Proceedings of the 13th International Middleware Conference*, Middleware, pages 1–20, New York, NY, USA, 2012. Springer-Verlag New York, Inc.

[40] Q. Dong and W. Dargie. Evaluation of the Reliability of RSSI for Indoor Localization. In *International Conference on Wireless Communications in Unusual and Confined Areas*, ICWCUCA, pages 1–6, Aug 2012.

[41] Y. Dong, D. Wang, N. Pissinou, and J. Wang. Multi-Path Load Balancing in Transport Layer. In *3rd Conference on Next Generation Internet Networks*, EuroNGI, pages 135–142, May 2007.

[42] T. Dreibholz, M. Becke, E. Rathgeb, and M. Tuxen. On the Use of Concurrent Multipath Transfer over Asymmetric Paths. In *IEEE Global Telecommunications Conference*, GLOBECOM, pages 1–6, Dec 2010.

[43] T. Dreibholz, R. Seggelmann, M. Txen, and E. P. Rathgeb. Transmission Scheduling Optimizations for Concurrent Multipath Transfer. In *Proceedings of the 8th International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports*, volume 8 of *PFLDNeT*, Lancaster, Pennsylvania/U.S.A., nov 2010.

[44] A. Dul. Global IP Network Mobility using Border Gateway Protocol. White Paper, Boeing, 2006. [Online; Accessed: 2015-07-31].

[45] A. Dutta, S. Madhani, W. Chen, O. Altintas, and H. Schulzrinne. Fast-handoff schemes for application layer mobility management. In *15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 3 of *PIMRC*, pages 1527–1532 Vol.3, Sept 2004.

[46] A. H. Eden. Three Paradigms of Computer Science. *Minds Mach.*, 17(2):135–167, July 2007.

[47] P. Edwards. Trusted Mobile Platform. http://www.trump-india-uk.org/. [Online; Accessed: 2015-07-31].

[48] E. Efstathiou, P. Frangoudis, and G. Polyzos. Controlled Wi-Fi Sharing in Cities: A Decentralized Approach Relying on Indirect Reciprocity. *IEEE Transactions on Mobile Computing*, 9(8):1147–1160, Aug 2010.

[49] P. Engelstad, A. Tonnesen, A. Hafslund, and G. Egeland. Internet Connectivity for Multi-homed Proactive Ad-Hoc Networks. In *IEEE International Conference on Communications*, volume 7, pages 4050–4056, June 2004.

[50] M. Ergen and A. Puri. MEWLANA-Mobile IP Enriched Wireless Local Area Network Architecture. In *Proceedings of the 56th IEEE Conference on Vehicular Technology*, volume 4 of *VTC-Fall*, pages 2449–2453, 2002.

[51] C. A. EU. Tasker: Total Automation for Android. http://tasker.dinglisch.net/. [Online; Accessed: 2015-11-11].

[52] K. Evensen. *Aggregating the Bandwidth of Multiple Network Interfaces to Increase the Performance of Networked Applications*. PhD thesis, University of Oslo, 2012.

[53] K. Evensen, D. Kaspar, P. Engelstad, A. Hansen, C. Griwodz, and P. Halvorsen. A network-layer proxy for bandwidth aggregation and reduction of IP packet reordering. In *IEEE 34th Conference on Local Computer Networks.*, LCN, pages 585–592, Oct 2009.

[54] K. Evensen, D. Kaspar, C. Griwodz, P. Halvorsen, A. Hansen, and P. Engelstad. Improving the Performance of Quality-adaptive Video Streaming over Multiple Heterogeneous Access Networks. In *Proceedings of the 2nd Annual ACM Conference on Multimedia Systems*, MMSys, pages 57–68, New York, NY, USA, 2011. ACM.

[55] K. Evensen, D. Kaspar, A. Hansen, C. Griwodz, and P. Halvorsen. Using Multiple Links to Increase the Performance of Bandwidth-Intensive UDP-based Applications. In *IEEE Symposium on Computers and Communications*, ISCC, pages 1117–1122, June 2011.

[56] X. Fan, H. Feng, and M. Yuan. PCA based on mutual information for acoustic environment classification. In *Audio, Language and Image Processing (ICALIP), 2012 International Conference on*, pages 270–275, July 2012.

[57] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. The Locator/ID Separation Protocol (LISP). RFC 6830 (Experimental), January 2013.

[58] D. Farinacci, D. Lewis, D. Meyer, and C. White. LISP Mobile Node. IETF - Draft, January 2015.

[59] L. Fon Wireless. Fon. `https://corp.fon.com/en`. [Online; Accessed: 2015-07-31].

[60] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824 (Experimental), January 2013.

[61] R. Fracchia, C. Casetti, C. Chiasserini, and M. Meo. WiSE: Best-Path Selection in Wireless Multihoming Environments. *IEEE Transactions on Mobile Computing*, 6(10):1130–1141, Oct 2007.

[62] Freedesktop.org. gstreamer: Open Source Multimedia Framework. http://gstreamer.freedesktop.org/. [Online; Accessed: 2015-11-11].

[63] V. Fuller and D. Farinacci. Locator/ID Separation Protocol (LISP) Map-Server Interface. RFC 6833 (Experimental), January 2013.

[64] L. Gao, G. Iosifidis, J. Huang, and L. Tassiulas. Hybrid Data Pricing for Network-assisted User-provided Connectivity. In *Proceedings of the 31st IEEE International Conference on Computer Communications*, INFOCOM, pages 682–690, April 2014.

[65] P. Georgopoulos, B. McCarthy, and C. Edwards. A Collaborative AAA Architecture to Enable Secure Real-World Network Mobility. In J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, and C. Scoglio, editors, *NETWORKING*, volume 6640 of *Lecture Notes in Computer Science*, pages 212–226. Springer Berlin Heidelberg, 2011.

[66] E. Goldoni and M. Schivi. End-to-End Available Bandwidth Estimation Tools, An Experimental Comparison. In F. Ricciato, M. Mellia, and E. Biersack, editors, *Traffic Monitoring and Analysis*, volume 6003 of *Lecture Notes in Computer Science*, pages 171–182. Springer Berlin Heidelberg, 2010.

[67] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil. Proxy Mobile IPv6. RFC 5213 (Proposed Standard), August 2008. Updated by RFC 6543.

[68] E. Gustafsson and A. Jonsson. Always best connected. *IEEE Wireless Communications*, 10(1):49–55, Feb 2003.

[69] S. Ha, I. Rhee, and L. Xu. CUBIC: A New TCP-friendly High-speed TCP Variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, Jul 2008.

[70] K. Habak, M. Youssef, and K. Harras. An Optimal Deployable Bandwidth Aggregation System. *Computer Networks*, 57(15):3067–3080, 2013.

[71] B. Han, P. Hui, V. Kumar, M. Marathe, J. Shao, and A. Srinivasan. Mobile Data Offloading Through Opportunistic Communications and Social Participation. *IEEE Transactions on Mobile Computing*, 11(5):821–834, May 2012.

[72] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan. Cellular Traffic Offloading Through Opportunistic Communications: A Case Study. In *Proceedings of the 5th ACM Workshop on Challenged Networks*, CHANTS, pages 31–38, New York, NY, USA, 2010. ACM.

[73] O. Helgason, E. Yavuz, S. Kouyoumdjieva, L. Pajevic, and G. Karlsson. A Mobile Peer-to-peer System for Opportunistic Content-centric Networking. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Networking, Systems, and Applications on Mobile Handhelds*, MobiHeld, pages 21–26, New York, NY, USA, 2010. ACM.

[74] H. Hsieh and R. Sivakumar. A Transport Layer Approach for Achieving Aggregate Bandwidths on Multi-homed Mobile Hosts. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, MobiCom, pages 83–94, New York, NY, USA, 2002. ACM.

[75] C. Hsu and U. Kremer. IPERF: A Framework for Automatic Construction of Performance Prediction Models. In *IN WORKSHOP ON PROFILE AND FEEDBACK-DIRECTED COMPILATION*, PFDC, 1998.

[76] N. Hu and P. Steenkiste. Evaluation and Characterization of Available Bandwidth Probing Techniques. *IEEE J.Sel. A. Commun.*, 21(6):879–894, Sep 2006.

[77] W. Hu and G. Cao. Energy Optimization Through Traffic Aggregation in Wireless Networks. In *Proceedings of the IEEE Conference on Computer Communications*, INFOCOM, pages 916–924, April 2014.

[78] X. Hu, L. Li, Z. Mao, and Y. Yang. Wide-Area IP Network Mobility. In *27th IEEE International Conference on Computer Communications*, INFOCOM, April 2008.

[79] C. Huang, K. Lan, and C. Tsai. A Survey of Opportunistic Networks. In *22nd International Conference on Advanced Information Networking and Applications - Workshops*, AINAW, pages 1672–1677, March 2008.

[80] hyperrealm. libconfig. http://www.hyperrealm.com/libconfig/. [Online; Accessed: 2015-11-11].

[81] IEEE Standard for Local and Metropolitan Area Networks–Link Aggregation. *IEEE Std 802.1AX-2008*, pages 1–163, Nov 2008.

[82] M. Integrated. DS2438 Smart Battery Monitor - Data Sheet. http://datasheets.maximintegrated.com/en/ds/DS2438.pdf. [Online; Accessed: 2015-11-11].

[83] J. Ioannidis, D. Duchamp, and G. Maguire, Jr. IP-based Protocols for Mobile Internetworking. In *Proceedings of the Conference on Communications Architecture & Protocols*, SIGCOMM, pages 235–245, New York, NY, USA, 1991. ACM.

[84] G. Iosifidis and I. Koutsopoulos. Double auction mechanisms for resource allocation in autonomous networks. *IEEE Journal on Selected Areas in Communications*, 28(1):95–102, January 2010.

[85] iPass. Q3 iPass Mobile Workforce Report, September 2013. Accessed: 2015-07-31.

[86] D. Johansen, H. Johansen, T. Aarflot, J. Hurley, Å. Kvalnes, C. Gurrin, S. Zav, B. Olstad, E. Aaberg, H. R. T. Endestad, C. Griwidz, and P. Halvorsen. DAVVI: A Prototype for the Next Generation Multimedia Entertainment Platform. In *Proceedings of the 17th ACM International Conference on Multimedia*, MM, pages 989–990, New York, NY, USA, 2009. ACM.

[87] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775 (Proposed Standard), June 2004. Obsoleted by RFC 6275.

[88] I. T. Jolliffe. Principal component analysis. Hardcover, October 2002.

[89] E. Jung, Y. Wang, I. Prilepov, F. Maker, X. Liu, and V. Akella. User-profile-driven Collaborative Bandwidth Sharing on Mobile Phones. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing; Services: Social Networks and Beyond*, MCS, pages 2:1–2:9, New York, NY, USA, 2010. ACM.

[90] G. Kang, J. Liu, M. Tang, and B. Cao. Web Service Selection Algorithm Based on Principal Component Analysis. *Journal of Electronics*, 30(2):204–212, 2013.

[91] S. Kang and J. Kim. QoS-Aware Path Selection for Multi-Homed Mobile Terminals in Heterogeneous Wireless Networks. In *7th IEEE Conference on Consumer Communications and Networking*, CCNC, pages 1–2, Jan 2010.

[92] I. Karma Mobility. Karma. `http://www.yourkarma.com`. [Online; Accessed: 2015-07-31].

[93] S. Kashihara, T. Nishiyama, K. Iida, H. Koga, Y. Kadobayashi, and S. Yamaguchi. Path selection using active measurement in multi-homed wireless networks. In *Proceedings of the International Symposium on Applications and the Internet.*, pages 273–276, 2004.

[94] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou. MicroCast: Cooperative Video Streaming on Smartphones. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys, pages 57–70, New York, NY, USA, 2012. ACM.

[95] M. Keller-Ressel. Lyapunov Function. `http://mathworld.wolfram.com/LyapunovFunction.html`. [Online; Accessed: 2015-07-31].

[96] F. Kelly, A. Maulloo, and D. Tan. Rate Control in Communication Networks: Shadow Prices, Proportional Fairness and Stability. In *Journal of the Operational Research Society*, volume 49, 1998.

[97] F. Kelly and T. Voice. Stability of End-to-end Algorithms for Joint Routing and Rate Control. *SIGCOMM Comput. Commun. Rev.*, 35(2):5–12, Apr 2005.

[98] M. Khalili, L. Gao, J. Huang, and B. Khalaj. Incentive Design and Market Evolution of Mobile User-Provided Networks. *CoRR*, abs/1502.06327, 2015.

[99] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec. MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution. *IEEE/ACM Transactions on Networking*, 21(5):1651–1665, Oct 2013.

[100] Q. Le-Trung, P. Engelstad, T. Skeie, and A. Taherkordi. Load-balance of Intra/inter-MANET Traffic over Multiple Internet Gateways. In *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia*, MoMM, pages 50–57, New York, NY, USA, 2008. ACM.

[101] K. Leung, G. Dommety, V. Narayanan, and A. Petrescu. Network Mobility (NEMO) Extensions for Mobile IPv4. RFC 5177 (Proposed Standard), April 2008. Updated by RFC 6626.

[102] D. Lewis, D. Meyer, D. Farinacci, and V. Fuller. Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites. RFC 6832 (Experimental), January 2013.

[103] J. Liao, J. Wang, and X. Zhu. cmpSCTP: An Extension of SCTP to Support Concurrent Multi-Path Transfer. In *IEEE International Conference on Communications*, ICC, pages 5762–5766, May 2008.

[104] J. Liu and S. Chung. An Efficient Load Balancing Scheme for Multi-Gateways in Wireless Mesh Networks. *Journal of Information Processing Systems*, 9(3):365–378, 2011.

[105] I. M87. M87. `http://www.m-87.com`. [Online; Accessed: 2015-07-31].

[106] M. Mace. The Truth about the Wireless Bandwidth Crisis. `http://mobileopportunity.blogspot.co.uk/2011/06/truth-about-wireless-bandwidth-crisis.html`. [Online; Accessed: 2015-07-31].

[107] J. Malinen. WPA Supplicant. `https://w1.fi/wpa_supplicant/`. [Online; Accessed: 2015-11-11].

[108] G. Mathews. On the Partition of Numbers. *Proceedings of the London Mathematical Society*, s1-28(1):486–490, 1896.

[109] P. McCann. Make-Before-Break Handoffs with Mobile IPv4. IETF - Draft, April 2008.

[110] B. McCarthy, M. Jakeman, and C. Edwards. Supporting Nested NEMO networks with the Unified MANEMO Architecture. In *IEEE 34th Conference on Local Computer Networks*, LCN, pages 609–616. IEEE, 2009.

[111] J. Millman. NumPy. `http://github.com/numpy/numpy`. [Online; Accessed: 2015-11-11].

[112] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol. RFC 5201 (Experimental), April 2008. Obsoleted by RFC 7401, updated by RFC 6253.

[113] P. Natarajan, N. Ekiz, P. Amer, J. Iyengar, and R. Stewart. Concurrent Multipath Transfer Using SCTP Multihoming: Introducing the Potentially-Failed Destination State. In *Networking*, pages 727–734. Springer, 2008.

[114] J. Navratil and R. Cottrell. ABwE: A Practical Approach to Available Bandwidth Estimation. In *Proceedings of the Workshop on Passive and Active Measurement*, PAM, 2003.

[115] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich. Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.). IETF - Draft, April 2008.

[116] Q. Nguyen-Vuong, N. Agoulmine, and Y. Ghamri-Doudane. A User-centric and Context-aware Solution to Interface Management and Access Network Selection in Heterogeneous Wireless Environments. *Computer Networks*, 52(18):3358–3372, Dec 2008.

[117] E. Nordmark and M. Bagnulo. Shim6: Level 3 Multihoming Shim Protocol for IPv6. RFC 5533 (Proposed Standard), June 2009.

[118] ns 3 developers. Network Simulator 3. https://www.nsnam.org/. [Online; Accessed: 2015-11-11].

[119] Ofcom. Implementing TV White Spaces, February 2015. Accessed: 2015-07-31.

[120] E. Ong, J. Kneckt, O. Alanen, C. Zheng, T. Huovinen, and T. Nihtila. IEEE 802.11ac: Enhancements for Very High Throughput WLANs. In *IEEE 22nd International Symposium on Personal Indoor and Mobile Radio Communications*, PIMRC, pages 849–853, Sept 2011.

[121] C. Paasch. *Improving Multipath TCP*. PhD thesis, UCLouvain / ICTEAM / EPL, November 2014.

[122] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. Exploring Mobile/WiFi Handover with Multipath TCP. In *Proceedings of the ACM SIGCOMM workshop on Cellular Networks: Operations, Challenges, and Future Design*, CellNet, pages 31–36, New York, NY, USA, 2012. IEE.

[123] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. Exploring Mobile/WiFi Handover with Multipath TCP. In *Proceedings of the ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design*, CellNet, pages 31–36, New York, NY, USA, 2012. ACM.

[124] C. Paasch, R. Khalili, and O. Bonaventure. On the Benefits of Applying Experimental Design to Improve Multipath TCP. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT, pages 393–398, New York, NY, USA, 2013. ACM.

[125] C. Perkins. IP Mobility Support for IPv4. RFC 3344 (Proposed Standard), August 2002. Obsoleted by RFC 5944, updated by RFCs 4636, 4721.

[126] C. Perkins. IP Mobility Support for IPv4, Revised. RFC 5944 (Proposed Standard), November 2010.

[127] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[128] C. Perkins and D. B. Johnson. Route Optimization in Mobile IP. IETF - Draft, September 2001.

[129] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, October 1994.

[130] A. Petz, A. Lindgren, P. Hui, and C. Julien. MADServer: A Server Architecture for Mobile Advanced Delivery. In *Proceedings of the Seventh ACM International Workshop on Challenged Networks*, CHANTS, pages 17–22, New York, NY, USA, 2012. ACM.

[131] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The Design and Implementation of Open vSwitch. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, NSDI, pages 117–130, Berkeley, CA, USA, 2015. USENIX Association.

[132] D. S. Phatak, T. Goff, and J. Plusquellic. IP-in-IP Tunneling to Enable the Simultaneous Use of Multiple IP Interfaces for Network Level Connection Striping. *Comput. Netw.*, 43(6):787–804, Dec 2003.

[133] L. Qi, W. Dou, and J. Chen. Weighted Principal Component Analysis-Based Service Selection Method for Multimedia Services in Cloud. *Springer, Journal of Computing*, pages 1–20, 2014.

[134] L. Qi, W. Dou, and J. Chen. Weighted Principal Component Analysis-based Service Selection Method for Multimedia Services in Cloud. *Springer Journal of Computing*, pages 1–20, 2014.

[135] C. Raiciu, M. Handley, and D. Wischik. Coupled Congestion Control for Multipath Transport Protocols. RFC 6356 (Experimental), October 2011.

[136] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *9th USENIX Symposium on Networked Systems Design and Implementation*, NSDI, pages 399–412, San Jose, CA, 2012. USENIX.

[137] C. Raiciu, D. Niculescu, M. Bagnulo, and M. J. Handley. Opportunistic Mobility with Multipath TCP. In *Proceedings of the Sixth International Workshop on Mobility in the Evolving Internet Architecture*, MobiArch, pages 7–12, New York, NY, USA, 2011. ACM.

[138] K. Ramachandran. Mobile IP Deployment After a Decade. White Paper, 2006.

[139] R. Ramjee, K. Varadhan, L. Salgarelli, S. Thuel, S. Wang, and T. La Porta. HAWAII: A Domain-based Approach for Supporting Mobility in Wide Area Wireless Networks. *IEEE/ACM Transactions on Networking*, 10(3):396–410, Jun 2002.

[140] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1654 (Proposed Standard), July 1994. Obsoleted by RFC 1771.

[141] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006. Updated by RFCs 6286, 6608, 6793, 7606, 7607.

[142] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. MAR: A Commuter Router Infrastructure for the Mobile Internet. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*, MobiSys, pages 217–230, New York, NY, USA, 2004. ACM.

[143] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141, 6665, 6878, 7462, 7463.

[144] P. Ruiz and A. Gmez-Skarmeta. Adaptive Gateway Discovery Mechanisms to Enhance Internet Connectivity for Mobile Ad Hoc Networks. *Ad Hoc and Sensor Wireless Networks*, 1(1-2), 2005.

[145] M. Santos, B. De Oliveira, C. Margi, B. Nunes, T. Turletti, and K. Obraczka. Software-Defined Networking Based Capacity Sharing in Hybrid Networks. In *21st IEEE International Conference on Network Protocols*, ICNP, pages 1–6, Oct 2013.

[146] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (INTERNET STANDARD), July 2003. Updated by RFCs 5506, 5761, 6051, 6222, 7022, 7160, 7164.

[147] S. Sevilla and J. Garcia-Luna-Aceves. HIDRA: Hiding Mobility, Multiplexing, and Multi-homing from Internet Applications. In *33rd IEEE Conference on Computer Communications, Workshops*, INFOCOM, pages 73–78, April 2014.

[148] R. Shacham, H. Schulzrinne, S. Thakolsri, and W. Kellerer. Session Initiation Protocol (SIP) Session Mobility. RFC 5631 (Informational), October 2009.

[149] A. Shahid and K. Humayun. Hybrid Scheme for Discovering and Selecting Internet Gateway in Mobile Ad-Hoc Network. *International Journal of Wireless and Mobile Networks*, 3(4), August 2011.

[150] C. Shannon. Communication In The Presence Of Noise. volume 86, pages 447–457, Feb 1998.

[151] P. Sharma, S. Lee, J. Brassil, and K. Shin. Handheld Routers: Intelligent Bandwidth Aggregation for Mobile Collaborative Communities. In *Proceedings of the 1st International Conference on Broadband Networks*, BroadNets, pages 537–547, Oct 2004.

[152] E. Shechtman and M. Irani. Matching Local Self-Similarities across Images and Videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, pages 1–8, June 2007.

[153] J. Shin, H. Lee, J. Na, A. Park, and S. Kim. Load Balancing Among Internet Gateways in Ad-Hoc Networks. In *62nd IEEE Vehicular Technology Conference*, volume 3 of *VTC*, pages 1677–1680, Sept 2005.

[154] A. Singh, S. Chakraborty, and T. K. Roy. Village Size in India. volume 4, pages 111–134, 2008.

[155] V. Singh, S. Ahsan, and J. Ott. MPRTP: Multipath Considerations for Real-time Media. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys, pages 190–201, New York, NY, USA, 2013. ACM.

[156] H. Sivakumar, S. Bailey, and R. Grossman. PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks. In *ACM/IEEE Conference on Supercomputing*, pages 38–38, Nov 2000.

[157] K. Sklower, B. Lloyd, G. McGregor, D. Carr, and T. Coradetti. The PPP Multilink Protocol (MP). RFC 1990 (Draft Standard), August 1996.

[158] A. Snoeren. Adaptive Inverse Multiplexing for Wide-Area Wireless Networks. In *Global Telecommunications Conference*, volume 3 of *GLOBECOM*, pages 1665–1672 vol.3, 1999.

[159] Q. Song and A. Jamalipour. Network selection in an Integrated Wireless LAN and UMTS Environment Using Mathematical Modeling and Computing Techniques. *Wireless Communications, IEEE*, 12(3):42–48, June 2005.

[160] W. Stevens and M. Thomas. Advanced Sockets API for IPv6. RFC 2292 (Informational), February 1998. Obsoleted by RFC 3542.

[161] R. Stewart. Stream Control Transmission Protocol. RFC 4960 (Proposed Standard), September 2007. Updated by RFCs 6096, 6335, 7053.

[162] M. Stiemerling. A System for Peer-to-Peer Video Streaming in Resource Constrained Mobile Environments. In *Proceedings of the ACM Workshop on User-provided Networking*, U-NET, 2009.

[163] M. Stiemerling. *Cooperative Internet Access in Resource Constrained Environments*. PhD thesis, University of Gttingen, 2011.

[164] R. Suoranta and A. Lappetelinen. Operators Dilemma - How to take advantage of the growing mobile Internet. `http://owni.fr/files/2011/09/Internet_growth_V10.pdf`. [Online; Accessed: 2015-07-31].

[165] Y. sup Lim, Y.-C. Chen, E. Nahum, D. Towsley, and K.-W. Lee. Cross-layer Path Management in Multi-path Transport Protocol for Mobile Devices. In *IEEE Conference on Computer Communications*, INFOCOM, pages 1815–1823, April 2014.

[166] D. Syrivelis, G. Iosifidis, D. Delimpasis, K. Chounos, T. Korakis, and L. Tassiulas. Bits and Coins Supporting Collaborative Consumption of Mobile Internet. In *Proceedings of the 34th IEEE International Conference on Computer Communications*, INFOCOM, April 2015.

[167] C. Taylor. Cisco: The Future of Mobile Networks, Feb 2013.

[168] G. Tsirtsis, H. Soliman, N. Montavont, G. Giaretta, and K. Kuladinithi. Flow Bindings in Mobile IPv6 and Network Mobility (NEMO) Basic Support. RFC 6089 (Proposed Standard), January 2011.

[169] A. Walid, Q. Peng, J. Hwang, and S. Low. Balanced Linked Adaptation Congestion Control Algorithm for MPTCP. IETF - Draft, January 2016.

[170] B. Wang, W. Wei, J. Kurose, D. Towsley, K. R. Pattipati, Z. Guo, and Z. Peng. Application-Layer Multipath Data Transfer via TCP: Schemes and Performance Tradeoffs. *Performance Evaluation*, 64:965–977, 2007.

[171] J. Wannstrom. LTE-Advanced, June 2013. Accessed: 2015-07-31.

[172] A. Wilson, A. Lenaghan, and R. Malyan. Optimising Wireless Access Network Selection to Maintain QoS in Heterogeneous Wireless Environments. In L. Heinzl and N. Prasad, editors, *8th International Symposium on Wireless Personal Multimedia Communications*, WPMC, pages 1236–1240, Tokyo, Japan, Sep 2005. NICT.

[173] R. Winter and A. Ripke. Multipath TCP Support for Single-homed Endsystems. IETF - Draft RFC, February 2013. `http://tools.ietf.org/html/draft-wr-mptcp-single-homed-05`.

[174] D. Wischik, M. Handley, and M. B. Braun. The Resource Pooling Principle. *SIGCOMM Comput. Commun. Rev.*, 38(5):47–52, Sep 2008.

[175] R. Withnell. GitHub Repository. `http://github.com/richardwithnell/`. [Online; Accessed: 2015-07-31].

[176] R. Withnell and C. Edwards. Multipath Dissemination for Collaborative Mobile Internet Access. In *Workshop on Cellular Offloading to Opportunistic Networks, CARTOON*, IEEE International Conference on Mobile Ad hoc and Sensor Systems, MASS. IEEE, 2014.

[177] R. Withnell and C. Edwards. Towards a Context Aware Multipath-TCP. In *IEEE 40th Conference on Local Computer Networks*, LCN, pages 434–437, October 2015.

[178] Z. Ye, S. Krishnamurthy, and S. Tripathi. Effects of multipath routing on TCP performance in ad hoc networks. In *IEEE Global Telecommunications Conference*, volume 6 of *GLOBECOM*, pages 4125–4131, Nov 2004.

[179] A. Yi Ding, P. Hui, M. Kojo, and S. Tarkoma. Enabling Energy-aware Mobile Data Offloading for Smartphones Through Vertical Collaboration. In *Proceedings of the ACM Conference on CoNEXT Student Workshop*, CoNEXT Student, pages 27–28, New York, NY, USA, 2012. ACM.

[180] H. Yokota, A. Idoue, T. Hasegawa, and T. Kato. Link Layer Assisted Mobile IP Fast Handoff Method over Wireless LAN Networks. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, MobiCom, pages 131–139, New York, NY, USA, 2002. ACM.

[181] T. Yu, Z. Zhou, D. Zhang, X. Wang, Y. Liu, and S. Lu. INDAPSON: An Incentive Data Plan Sharing System Based on Self-Organizing Network. In *Proceedings of the 33rd IEEE International Conference on Computer Communications*, INFOCOM, pages 1545–1553, April 2014.

[182] D. Zhang, R. Shinkuma, and N. B. Mandayam. Bandwidth Exchange: An Energy Conserving Incentive Mechanism for Cooperation. *IEEE Transactions on Wireless Communications*, 9(6):2055–2065, June 2010.

[183] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang. A Transport Layer Approach for Improving End-to-end Performance and Robustness Using Redundant Paths. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC, pages 8–8, Berkeley, CA, USA, 2004. USENIX Association.

[184] W. Zhang, Q. Wu, W. Yang, and H. Li. Reliable Multipath Transfer Scheduling Algorithm Research and Prototype Implementation. In *Proceedings of the Asia-Pacific Advanced Network*, pages 45–52. APAN, 2010.

[185] Z. Zhu, R. Wakikawa, and L. Zhang. A Survey of Mobility Support in the Internet. RFC 6301 (Informational), July 2011.

APPENDIX A

Mobile Connectivity

The Internet Protocol Suite has not fundamentally changed since its inception, and it is ingrained into current infrastructure and user devices. Originally neither end-host mobility nor multihoming were considered during the definition of IP and TCP. Due to the pervasive and dependable nature of the current protocols, introducing change at the core layers in the stack understandably meets resistance. This has made the necessary deployment of innovative mobile connectivity models more challenging, this resistance to change is evident in the lack of real world IPv6 usage. Pragmatically the goal of any new protocol proposal should be widespread adoption within its domain, therefore despite the numerous benefits that new and innovative protocols may bring, the real measurement of success is deployment, which will become evident throughout the remainder of this chapter and will be carried through to the design. Therefore protocols that are incrementally deployable and don't require substantial change to the current infrastructure, are more likely to see extensive uptake. The focus for this section, will be on the protocols and technologies that provide devices with connectivity while mobile, both globally and locally.

A.1 Terminology

There are a number of new terms that are exclusive to the mobility domain that will be used throughout the remainder of this section and are as follows:

Handover – Migration from one point of attachment to another.

Horizontal Handover – A handoff between two network access points that

use the same access technology (homogenous).

Vertical Handover – A handoff between two network access points that use different access technologies (heterogeneous).

Hard Handover – A handoff is forced by breaking the physical connection, for instance by moving out of range of an access point. (Break before make)

Soft Handover – A handoff utilizes multiple channels or access technologies in parallel, establishing the new connection before the old connection breaks. (Make before break)

Fast Handover – An approach to reducing handover delay when changing points of attachment.

Intradomain Mobility – Mobility (handover) between points of attachment within the same subnet.

Interdomain Mobility – Mobility (handover) between points of attachment across different networks.

A.2 Mobility

This section describes, protocols and technologies that enable the mobility of both hosts and networks. In this context, mobility is defined as the ability for a mobile device to move from one network to another, changing its point of attachment to the Internet, while seamlessly maintaining any connections or flows. Mobility protocols have been a significant area of research for a number of years. The core aim of this research has been to reduce the impact when a host moves from one network to another, this process is known as handover. The key measurement of handover is the length of time it takes for the connections to migrate from one point of attachment to another. The shorter the time frame, the smaller the impact the handover will have on the active applications. This can be measured not only in terms of time but additional metrics such as packet loss or the required number of packets that need to be retransmitted.

Mobility protocols have seen a significant level of interest since their inception. First proposed in [83], approaches have been revised and revisited countless

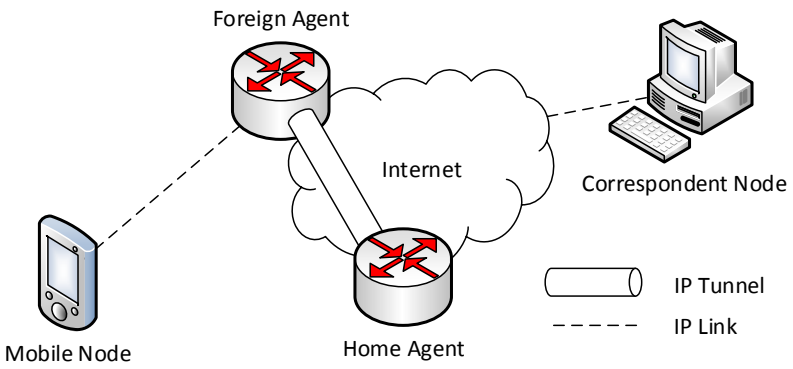


Figure A.1: An example of the architecure for Mobile IP.

times. Ranging from improving the speed of the handover using link layer information [180], to application layer intelligence aiming to minimise the impact that handover has on the user [45]. For the remainder of this section, key mobility protocols are first categorised, regarding the different types of mobility that exist.

There are two fundamental types of approach to supporting host mobility, firstly there are routing based approaches, in which a mobile host keeps its IP address and the routing system in the network accommodates for any changes in the point of attachment. Examples of route based mobility include, HAWAII [139] and Connexion [44]. The second approach relies on mapping, as described in [185], all mapping based mobility protocols rely on three components: an identifier for the mobile host, a locator for where the mobile host current resides and finally a mapping between these two components. Examples of mapping based mobility solutions include [126], [57], [8]. This mapping based approach typically introduces a layer of indirection as with Mobile IP [126], leading to inefficient triangular routing, an example of Mobile IP is illustrated in Fig. A.1. Due to the scalability issues with routing based mobility approaches as discussed in [185] for the remainder of this chapter we will only consider technologies that rely on mapping. Typically mapping based solutions possess the following components, while the terminology may change between implementations the functionality remains the same:

Mobile Node (MN) – The Mobile Node (mobile host) changes its point of attachment to the Internet, from the Home Network to a Foreign Network.

The Mobile Node sends all packets to the Home Agent, while Correspondent Nodes, send all packets destined for the Mobile Node to the Home IP.

Home Network (HN) – The Mobile Node has a permanent bidirectional tunnel to the home network.

Home Agent (HA) – The Home Agent is the host residing in the Home Network that is responsible for ensuring the reachability of the Mobile Node. This works by allocating the Mobile Node a fixed IP address that the Home Agent can use to communicate.

Foreign Network (FN) – The Foreign Network is any network aside from the Home Network, which the Mobile Node may connect to.

Foreign Agent (FA) – The Foreign Agent is the router or access point that provides the Mobile Node with connectivity while connected to a Foreign Network.

Care-of-Address (CoA) – The Care of Address, is provided to the Mobile Node by the foreign agent. This address is used by the Mobile Node to communicate with the Home Agent and setup the tunnelling and routing required.

Correspondent Node (CN) – The correspondent node is the destination that the Mobile Node wishes to communicate with via IP.

A.2.1 Network Layer

Mobile IP [126] is one of the earliest and standardized approaches to provide global host mobility. First proposed in 1994, it relies on all the previously discussed components the architecture of which can be seen in Fig. A.1. As a Mobile Node roams from the Home Network across a number of Foreign Networks, the Mobile Node retains the address provided by the Home Network. When the Mobile Node changes its point of attachment, connecting to a Foreign Network, the Foreign Agent provides the Mobile Node with a Care-of-Address which the Mobile Node subsequently registers with the Home Agent. To this end, the Home Agent is always able to route packets between the Mobile Node and any Correspondent Node. Routing all packets via the Home Agent introduces inefficient triangular routing, as all communication takes an indirect path. Triangular routing raises

a number of concerns in some mobile environments, if the Mobile Node resides a significant distance from the Home Agent the overhead incurred due to indirection can become significant, especially if attempting to support real time communications. For this reason, Route Optimization was proposed as an extension to Mobile IP [128]. Route Optimization allows a Correspondent Node or Mobile Node to tunnel their packets directly between one another, bypassing the Home Agent, improving the efficiency of routing. If the Correspondent Node is unaware of the Mobile Nodes current location packets are simply routed as normal via the Home Agent. The specification for Mobile IP was transformed for IPv6 [87], with inherent support for Route Optimization. Despite the functionality and performance optimizations provided by Mobile IP, it is still yet to see significant uptake or wide spread deployment. In [138] a number of deployment issues are discussed with the Mobile IP specification, which are undoubtedly significant as the protocol is still underutilized after 20 years. From a performance perspective, triangular routing is a significant limiting factor of the Mobile IP specification. Addressing this with Route Optimization requires significant change to all communicating hosts, needing to support both the tunnelling and mapping. Eliminating performance as a fundamental issue of the protocol still leaves economic and bureaucratic factors that make deployment challenging. For example, if a mobile user has a different service plan for both WiFi and Cellular connectivity, determining the provider responsible for the Home Agent and the subsequent agreements needed between providers is non-trivial and is typically not addressed by protocol implementations. If the Home Agent is created as a third party service, issues of trust and service layer agreements between providers can be simplified however it is likely that the user would be required to pay for an additional mobility plan to support this feature. Despite the lack of wide-spread Mobile IP deployment, it has been adopted within cellular networks. Proxy Mobile IPv6 (PMIP) is a popular extension to MIP, especially in cellular networks. PMIP enhances the mobility model by masking the address management from the Mobile Node, such that the entire handover process is handled by the network. To this end, a Mobile Access Gateway signals the binding update to the Home Agent (referred to as the Local Mobility Anchor)

Since the establishment of Mobile IP, a number of alternative network layer protocols have been proposed to support mobility [112] [57] [8] [67], as previously mentioned the architectures and approaches to all of these solutions remain

similar in terms of the required infrastructure and mapping between locators and identifiers. The Host Identity Protocol (HIP) specified in [112], attempts to bridge the gap between locators (IP addresses) and identifiers (Domain Name Service). HIP requires that cryptographic keys are used as host identifiers instead of IP addresses, therefore applications need to be modified to support this new paradigm. The use of cryptographic keys as an identifier allows hosts to authenticate changes in the locator, for improved security. To introduce naming into the HIP protocol, extensions have been proposed to the Domain Name System, which includes introducing a new Resource Record containing additional information including the hosts public key, which is required for authentication.

The Identifier-Locator Network Protocol (ILNP) [8] splits an IP address into a locator and an identifier. In the case of ILNPv6, the upper 64 bits are allocated to the Locator while the lower 64 bits are allocated as the Node Identifier. Therefore, end-to-end protocols such as TCP and UDP are expected to only use the Node Identifier, while the network layer subsequently uses the Locator. In comparison to IPv6 the ILNPv6 Locator matches the network prefix, while the Node Identifier matches the interface identifier. To further support multihoming aspects, a host may choose to use multiple Network Identifiers and Locators simultaneously. In ILNP dynamic mappings between the Node Identifier and Locator are supported, moreover a single Node Identifier can be mapped to multiple Locators, allowing a single transport layer end-point to be reachable over multiple networks at the same time. By updating these mappings handover is inherently supported. Furthermore, the ability to map multiple Locators to a single Node Identifier enables soft handover, allowing a make-before-break connectivity model. To support ILNP, as with HIP, new DNS Resource Records are needed. The new Resource Records includes an Identifier Record and a Locator Record. During communication with a Mobile Node, the Correspondent Node queries the DNS to obtain the current Locator. Additionally, the Mobile Node updates the DNS with new Locators as it changes its point of attachment. Finally the Mobile Node will send Locator Updates to the Correspondent Nodes, to update the set of current Locators. ILNP removes the need for triangular routing, simply relying on DNS updates and queries to enable the seamless communication between Mobile and Correspondent Nodes, which can provide a more efficient mobility model, than the previously discussed MIP implementations.

The Locator/Identifier Separation Protocol (LISP) [57] is fundamentally dif-

ferent to the Loc/ID split protocols discussed so far (HIP and ILNP). HIP and ILNP both introduce new name spaces, while LISP changes the underlying routing and addressing while being able to maintain IP addresses (either IPv4 or IPv6) which enables incremental deployment, as previously discussed this can be incredibly beneficial. The core contributions of LISP are to simplify routing and improve scalability. To this end, LISP defines two new name spaces to split the locators and identifiers, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs). LISP uses a distributed database service to map between EIDs and RLOCs. In the context of LISP, EIDs are assumed not to be globally routable, while RLOCs are; therefore, to forward a packet to the appropriate end-host LISP routers are expected to encapsulate packets that use an EID, with a header containing the corresponding RLOC. Consequently LISP is in fact better described as a map-and-encapsulate protocol. LISP Mobile Node (LISP-MN) is a specification for host mobility using LISP [58]. To support host mobility LISP-MN requires an implementation of LISP on the mobile host, a mapping server to update the RLOC [63] as the mobiles hosts point of attachment changes and Internetworking infrastructure to communicate with non-LISP hosts or networks[102].

A.2.2 Transport Layer

In the previous section, we introduced a number of resource pooling and bandwidth aggregation approaches that rely on hosts being multihomed to exploit path diversity in the Internet. This model can also be applied to mobility, as shown in [60] [147]. These novel techniques still rely on a mapping based approach as previously discussed, however as the mapping between the locator and identifier is stored at the end-hosts for a connection, additional infrastructure, in the way of the Home Agent or Mapping Server is not required. Typically these solutions reside above the network layer, relying on the connection between the two hosts to store the associated metadata, mapping multiple IP addresses to a single end-point. Mobile SCTP was first proposed in 2002, proposing that multihomed hosts could communicate with SCTP servers, while supporting seamless handover across any access technologies. Mobile SCTP exploits the ability for a connection to manage multiple IP addresses, when the connection is created the mobile host signals all of its available IP addresses to the server. As the list of IP addresses available to the mobile host changes, the server is signalled

with the corresponding updates. This end-host oriented model for mobile connectivity presented by Freeze TCP, TCP-Migrate and Mobile SCTP has become increasingly common among novel mobility models at the transport layer, such as MPTCP.

One of the implicit benefits of the MPTCP proposal is the inherent support for mobility [137]. As multiple subflows must be supported for a single connection end-point the identifiers and locators that are used are split. This split allows new MPTCP subflows to associate with the initial connection regardless of whether or not there is an active subflow. This decoupling between the subflows and the connections provides a number of benefits on top of Mobile IP and comparable network layer solutions. Typically Mobile IP requires the physical or link layer connectivity to change before a handover can occur from one network interface to another, known as break-before-make. With MPTCP it is no longer necessary to wait for a connection to break, as data for the same end-point can be transmitted simultaneously over multiple paths; a make-before-break model can be established, providing truly seamless handover as packets are shifted to a new path before the old path fails. Novel extensions and implementations to MIPv6 attempt to address these multihomed issues, by introducing Multiple Care-of-Addresses (MCoA) [168] and make-before-break connectivity [109]. Despite this, I believe that make-before-break is better suited to the transport layer, as the best the network layer can offer is redundancy, replicating packets across the set of available addresses during handover which must be handled before handing the packets up. One of the main drawbacks associated with MPTCP and this make-before-break model, it the increased power consumption that is required to constantly maintain flows over all available interfaces. As discussed in the previous section, MPTCP is flexible allowing links to be set to active or backup modes, additional handover modes have also been investigated [137]. Setting an interface to act as a backup or for single path can minimise the power consumption issues at the expense of a slight increase in handover delay. The key differentiator between single path and backup modes is when the subflows are created. For backup mode, backup subflows are created for each interface but data is not transmitted until all active subflows are no longer usable. While in single path mode, the subflows are only created once no other active subflows are available. The difference between full, backup and single path modes is presented in [123], in terms of throughput the full MPTCP mode performs best dropping from the

speed of WiFi to the speed of the 3G network, the backup mode experiences a larger drop (still not dropping to zero), while traffic migrates to the 3G path and the single path mode experiences the worst performance as throughput drops to zero, taking a few seconds for the interface to become active.

A.2.3 Application Layer

The Session Initiation Protocol (SIP) [143], is an application layer protocol describing how sessions are created, maintained and destroyed between two or more hosts. SIP is typically used for media applications, ranging from voice communications, to video conferencing. As users and devices became mobile, extensions were proposed to SIP to provide mobility, allowing the voice and video applications to maintain connectivity, supporting session migration between end-points. In [148] the authors present application layer mobility using SIP, this doesn't just include a device changing its IP address but also a user changing device or service provider. In this domain, application layer mobility is much better suited to meet the needs of the user than network or transport layer solutions, as a user may want to seamlessly migrate a call from a desktop or laptop, to a smart phone or tablet as the environment changes. Voice calls are arguably one of the most sensitive applications, in terms of the impact incurred by network latency and handover time. To this end, global solutions such as Mobile IP can limit the potential performance through triangular routing, the need for encapsulation, and the time required to update the home agent. SIP still retains a home agent like concept, referred to as the registrar which is updated as the host changes its IP address. Furthermore, during a SIP session, the mobile host simply informs the hosts associated with a session of any changes in IP address, minimizing handover delay.

HIDRA (Hidden Identifiers for Demultiplexing and Resolution Architecture) [147] is a recent novel proposal that changes how identifiers and locators are mapped. Effectively all of the discussed technologies thus far use what the authors describe as "open identifiers". Even though the identifiers are split or mapped to the appropriate locator, they still represent an attribute that is known outside of the end-hosts. The use of "open identifiers" is a fundamental problem in supporting the efficient and seamless evolution of the Internet as discussed in the start of this section. To this end, the author proposes "hidden identifiers" that

are agnostic of any network or transport technologies, allowing applications to remove their dependency on specific protocols. HIDRA is essentially an application layer solution to multihoming and mobility that changes the socket interface that an application uses, masking the complexity of the underlying networking. By introducing an additional layer of abstraction between the application and the network, mobility can be seamlessly supported as changes in IP address or TCP connections is transparent. Moreover, this abstraction can additionally allow new network and transport layer protocols to be deployed and tested without breaking or modifying applications that support HIDRA.

A.3 Mobile Networks

Individual hosts are not the only concern for mobility protocols, as mobile routers and co-located devices form mobile networks. In this scenario, host based mobility solutions may no longer be the most appropriate, as complex routes are formed at the edge, providing and extending connectivity. This area can be split into two parts; communication between these devices, typically supported by mobile ad-hoc protocols, and communication with remote hosts, provided by network mobility protocols. While these two areas can focus on fundamentally different goals, there is significant overlap between the two when considering real world deployments, as both challeneges need to be solved simultaneously.

A.3.1 Mobile Ad-Hoc

A Mobile Ad-Hoc Network (MANET) [31] is an infrastructure-less network connected by wireless links, in which the connected hosts are able to move independently and arbitrarily while dynamically routing data between one another. The mobility properties lead hosts to connect and reconnect to other hosts changing routes as links are removed or become available, automatically reconfiguring based on the current context. All the hosts in the MANET are expected to route traffic between one another, as well as out towards a gateway if available. There are a number of routing protocols that can be used in a MANET such as OLSR [32] or AODV[127]; the routing protocol chosen to implement a MANET is pivotal to the performance and functionality of the network. The IETF MANET working group [31] identified two key approaches, reactive (AODV) and proac-

tive (OLSR) routing to be considered in the mobile domain. Proactive MANET protocols periodically updates the routing tables of the hosts in the network even if no change has occurred, while reactive protocols only acquire current routing information as it is needed (when a packet needs to be routed). The reactive approach offers a reduced overhead in comparison to proactive protocols, but can take longer to obtain the required routes.

Ad hoc On-Demand Distance Vector (AODV) is one of the earliest standardised reactive MANET routing protocols. Directly connected hosts find neighbours through HELLO messages. If communication is needed between two non-neighbours, a Route Request is broadcast; if the recipients of the broadcast know the route it replies with a Route Reply, otherwise the Route Request is broadcast again. Optimized Link State Routing Protocol (OLSR) on the other hand, maintains routes as the state of the network changes, making sure they are available for use on demand. Similarly to AODV, OLSR uses HELLO messages to determine one-hop neighbours, and subsequently two-hop neighbours via the responses, however the HELLO messages are transmitted periodically as opposed to on-demand. OLSR then becomes more complex, as based on the set of two-hop neighbours, the hosts in the MANET elect a set of multipoint relays, responsible for relaying messages between hosts. Topology Control (TC) messages are then used to disseminate information about available neighbours to the rest of the hosts in the network. To ensure that all hosts have the same view of the network, OLSR frequently floods the network with the current topology. The Better Approach To Mobile Adhoc Networking (BATMAN) protocol [115] takes a different approach to both OLSR and AODV. While it is still a proactive protocol, the developers have proposed that the MANET functionality should be implemented at the link layer, as opposed to the network layer. Designing the protocol at the link layer creates a flat network topology, similar to that of a switch. BATMAN differs from the proactive OLSR by adopting a Distance Vector routing algorithm, instead of Link State, which reduces the amount of information each node is required to store about the topology. The Babel routing protocol [25] is a more recent attempt to address the problems of MANET deployments. Babel is based on other distance vector routing protocols, such as DSDV [129] and EIGRP [3]. Babel performs more intelligent route selection than the previously discussed protocols, by not only taking into account the number of hops, but also historical information, preferring routes that have been previously observed. Combining

the enhanced link selection with an improved implementation of the Estimated Transmission Count (ETX) algorithm [38], Babel is able to build a robust and efficient mesh network.

A.3.2 Network Mobility

Network mobility differs from the ad-hoc approach, typically considering how a set of connected hosts move together, maintaining connectivity as the mobile network as a whole changes it's point of attachment to the Internet. Typically solutions to this problem are similar to those proposed in the previous mobility section; focusing on routing based approaches, or using a mapping between a hosts location and its identifier. This problem becomes slightly more complex as it is effectively no longer the host that is mobile but the network. Connexion [44] was a proposal by Boeing, attempting to enable seamless connectivity to users travelling by air or sea. The authors propose the use of BGP for mobility, having a mobile gateway announce its prefix as it moves. This requires the network to be globally routable at all times, for a single aircraft, without frequent changes, this is potentially a feasible solution; however, the proposal lacks scalability and would have a significant impact on global routing tables. Wide-Area IP Network Mobility (WINMO) [78] is a more recent network mobility proposal, which uses a combination of routing and mapping. WINMO attempts to reduce the overhead of updating BGP to improve the scalability of the protocol. Similar to Mobile IP, WINMO extends the BGP updates with the concept of a home network and home agent, limiting and reducing the number of changes that must be made. The Network Mobility (NEMO) Working Group [101], have proposed a mobile network solution that is based on Mobile IP. For this to work, a Mobile Router is required which connects via a home agent through which all traffic is sent, as with Mobile IP while roaming the Mobile Router obtains a care of address which is used to keep the home agent up to date with its current location. If the Mobile Router is then required to route packets for the mobile network it informs the Home Agent by setting a flag in the binding update, this allows the home agent to forward packets destined for hosts within the network instead of just the Mobile Router. It is also possible for nesting to occur, allowing Mobile Routers to connect via another Mobile Router. This however leads to greater inefficiencies, each level of nesting can add an extra level of indirection through a new home agent leading to

"ping pong" routing between Home Agents. Similarly to Mobile IP, the NEMO protocol has been extended in recent years, introducing support for multihoming at different points in the mobile network. This multihomed connectivity model begins to adapt well to the problem domain, described in the previous chapter. As multiple users inter-connecting will often have additional ways of accessing the Internet. When a NEMO is multihomed, additional complexity is introduced as with nesting, as multiple Home Agents can be introduced, along with multiple Care-of-Addresses. Moreover, the NEMO may be multihomed from different hosts within the network. In addition to multihoming, NEMO can still encounter connectivity problems that are solved through the use of MANET protocols. To this end, the combination of both MANET and NEMO has been of significant interest to the Mobile community. This combination of technologies is typically referred to as MANEMO, which merges the flexible Ad-Hoc nature of MANET with the persistent global reachability of NEMO in order to take advantage of the beneficial characteristics that exist within both protocols. MANET protocols provide a flexible approach allowing Mobile Routers to communicate locally, while NEMO provides IP mobility enabling hosts to maintain their connections irrespective of any horizontal or vertical handover. A MANEMO is formed when multiple NEMO Mobile Routers connect together in an Ad-Hoc configuration, allowing Mobile Routers to route packets between one another using MANET routing. This MANEMO connectivity model can allow Mobile Routers to remain connected via a Home Agent, even in the event that the MANET separates locally. Despite the feature rich world of Network Mobility and specifically NEMO and MANEMO, the fundamental concept is based on Mobile IP, mapping and tunnelling, which as previously discussed has failed to become a core component in the world of mobile connectivity. While Network Mobility obviously still has a place in niche domains, such as Air travel, for the same reasons as the failure of Mobile IP, it does not present itself as the most appropriate technology for supporting cooperative and collaborative Internet access.

A.4 Summary

In this section, we have presented an overview of the current state of mobile connectivity models and architectures; including hosts, networks, interdomain

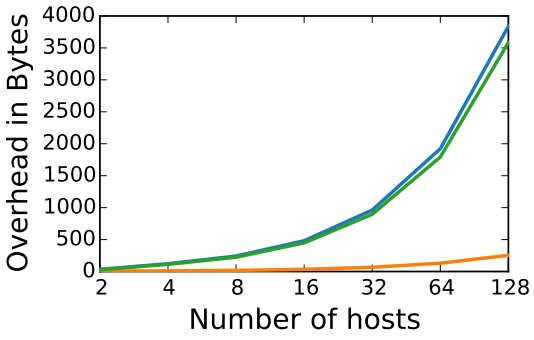
and intradomain mobility. As previously discussed, efficient and seamless mobility for both hosts and networks is still an open-ended problem, as standardised and accepted solutions do not encompass all necessary use cases for a mobile device. For example, Proxy Mobile IP is used within Cellular networks to maintain IP addresses, however this does not extend to vertical handover as users migrate to third-party WiFi access points. To this end, we believe mobility should be supported at each of the end-hosts, which are capable of establishing a complete view of available paths and connectivity options. This focus on end-host mobility fits well with transport and application layer mobility models, such as MPTCP and potentially HIDRA, which we believe to be much more appropriate for the future development of mobile network stacks, as they can additionally incorporate bandwidth aggregation, leveraging multihoming and resource pooling at the heart of the protocol.

APPENDIX B

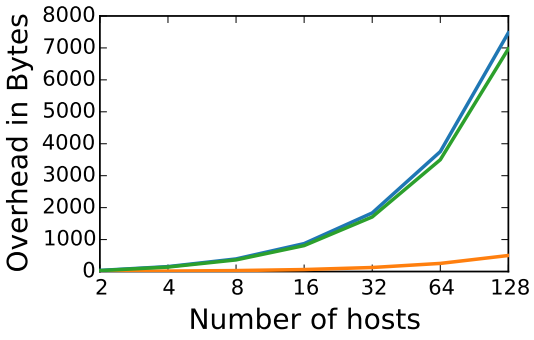
Additional Evaluation

In this section of the Appendix, we present additional results and graphs from the evaluation in Chapter 6.

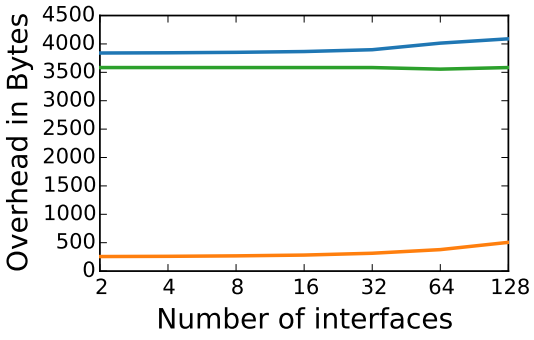
B.1 MAP Behaviour



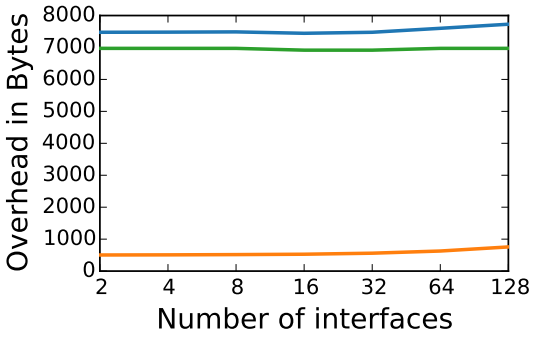
(a) Flat Network (gateways=1)



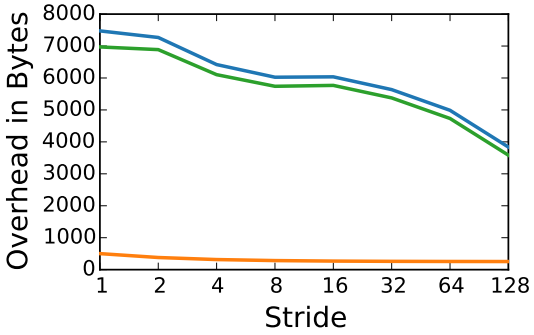
(b) Nested Network (gateways=1)

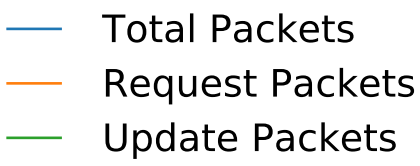


(c) Flat Network (hosts=128)



(d) Nested Network (hosts=128)





(e) Variable Stride (gateways=128, hosts=128)

Figure B.1: Overhead of running MAPD on various network topologies.

B.2 MAP Utilization

B.2.1 Flat Network

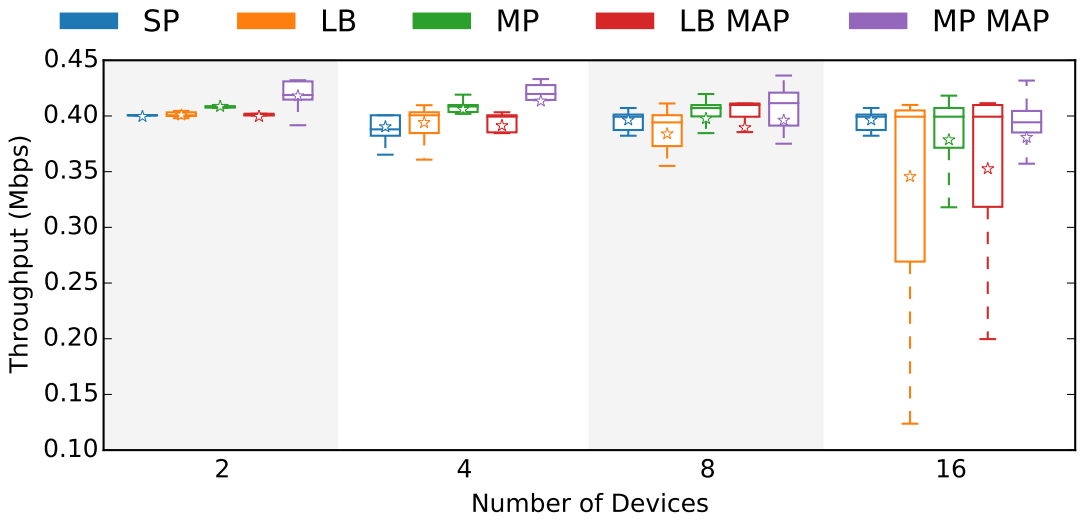


Figure B.2: Throughput from simulation of the flat topology for **64KB** flows using iPerf.

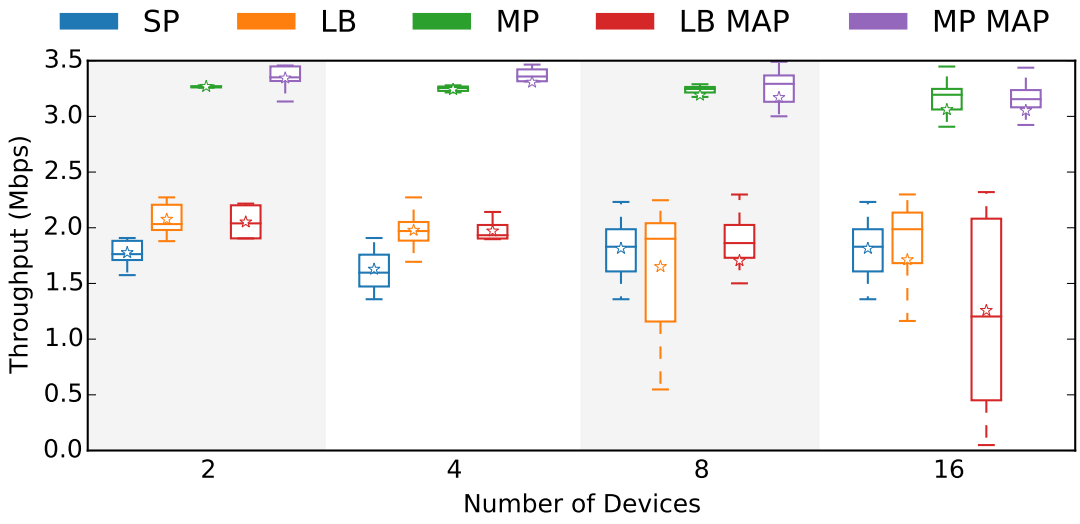


Figure B.3: Throughput from simulation of the flat topology for transferring **512KB** flows using iPerf.

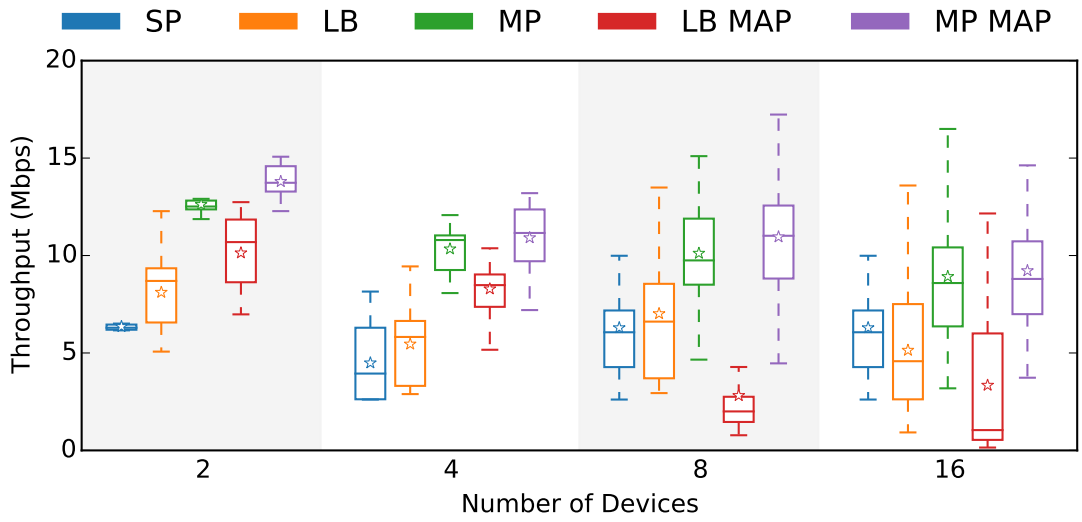


Figure B.4: Throughput from simulation of the flat topology for transferring **4MB** flows using iPerf.

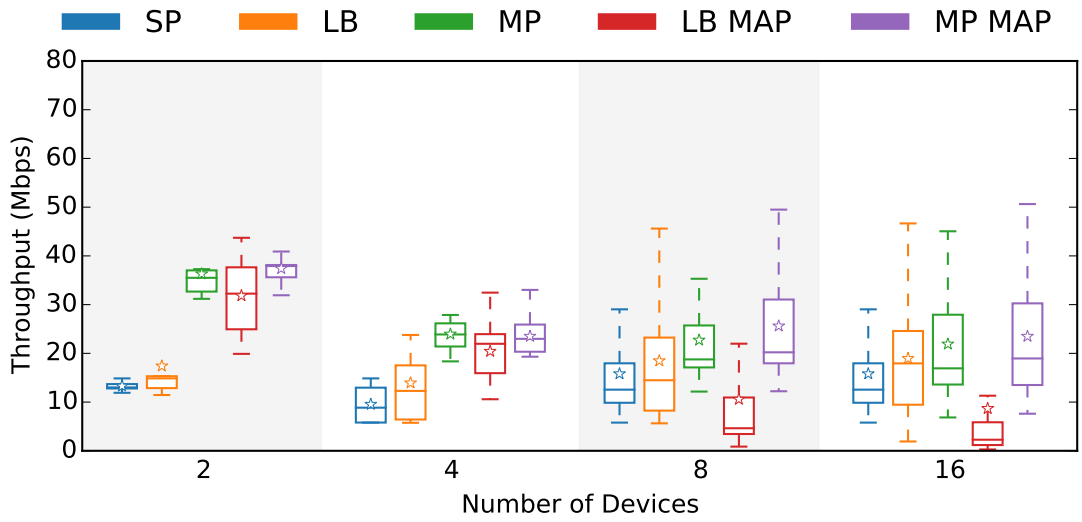


Figure B.5: Throughput from simulation of the flat topology for **32MB** flows using iPerf.

B.2.2 Single Host Access

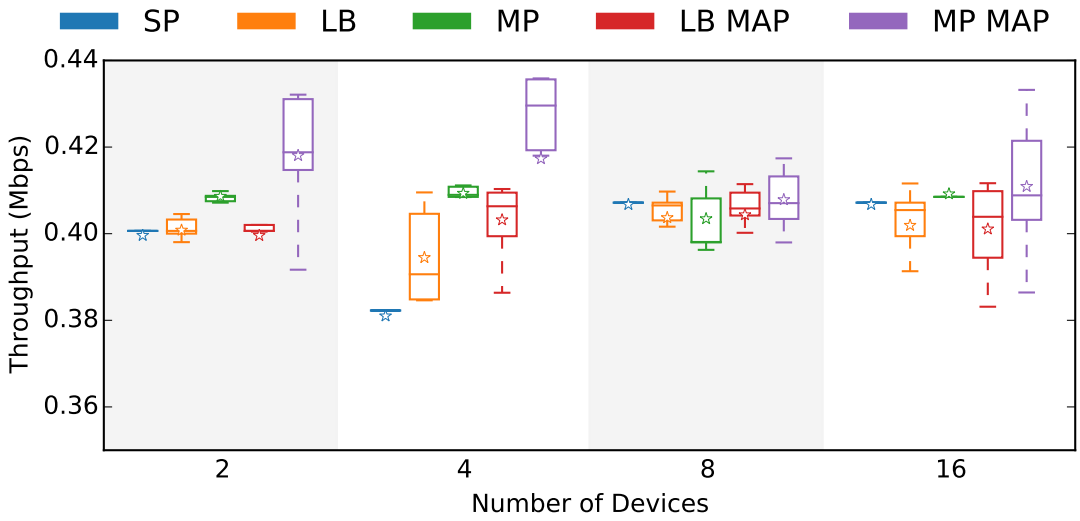


Figure B.6: Throughput from simulation of the nested topology, a single host creates **64KB** flows using iPerf.

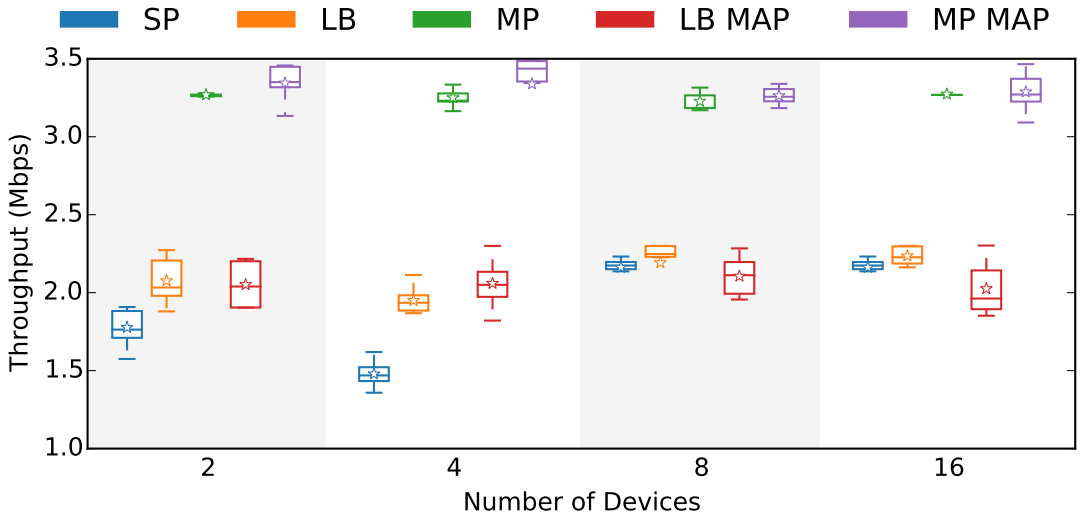


Figure B.7: Throughput from simulation of the nested topology, a single host creates **512KB** flows using iPerf.

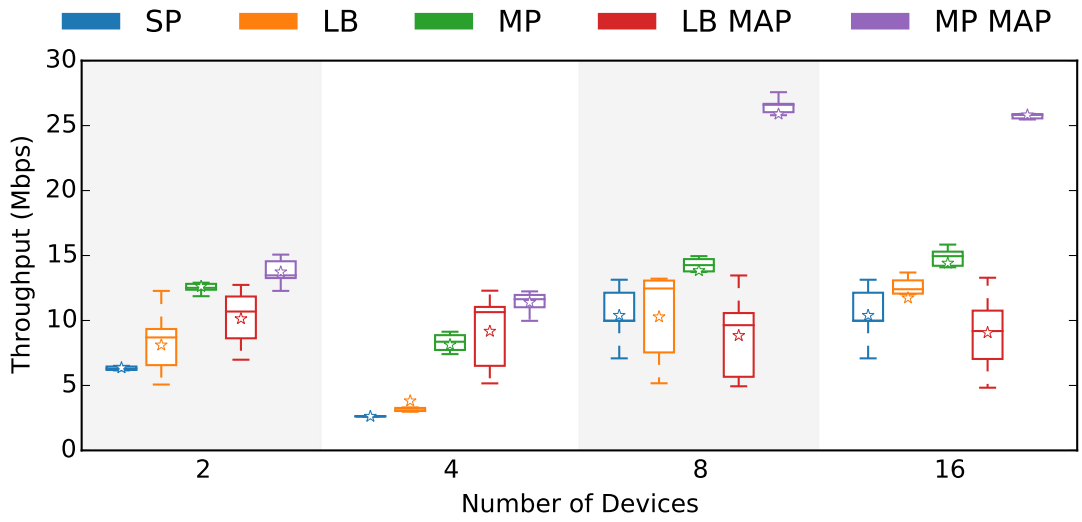


Figure B.8: Throughput from simulation of the nested topology, a single host creates **4MB** flows using iPerf.

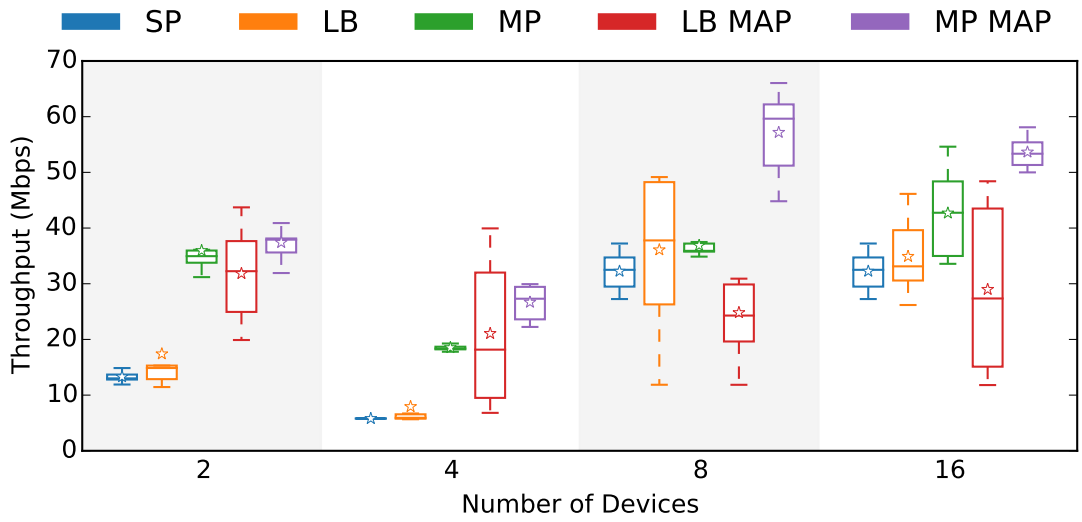


Figure B.9: Throughput from simulation of the nested topology, a single host creates **32MB** flows using iPerf.