Exploring Traffic and QoS Management Mechanisms to Support Mobile Cloud Computing using Service Localisation in Heterogeneous Environments

A thesis submitted to Middlesex University
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

Fragkiskos Sardis

School of Science and Technology
Middlesex University
August 2014

Abstract

In recent years, mobile devices have evolved to support an amalgam of multimedia applications and content. However, the small size of these devices poses a limit the amount of local computing resources. The emergence of Cloud technology has set the ground for an era of task offloading for mobile devices and we are now seeing the deployment of applications that make more extensive use of Cloud processing as a means of augmenting the capabilities of mobiles. Mobile Cloud Computing is the term used to describe the convergence of these technologies towards applications and mechanisms that offload tasks from mobile devices to the Cloud.

In order for mobile devices to access Cloud resources and successfully offload tasks there, a solution for constant and reliable connectivity is required. The proliferation of wireless technology ensures that networks are available almost everywhere in an urban environment and mobile devices can stay connected to a network at all times. However, user mobility is often the cause of intermittent connectivity that affects the performance of applications and ultimately degrades the user experience. 5th Generation Networks are introducing mechanisms that enable constant and reliable connectivity through seamless handovers between networks and provide the foundation for a tighter coupling between Cloud resources and mobiles.

This convergence of technologies creates new challenges in the areas of traffic management and QoS provisioning. The constant connectivity to and reliance of mobile devices on Cloud resources have the potential of creating large traffic flows between networks. Furthermore, depending on the type of application generating the traffic flow, very strict QoS may be required from the networks as suboptimal performance may severely degrade an application's functionality.

In this thesis, I propose a new service delivery framework, centred on the convergence of Mobile Cloud Computing and 5G networks for the purpose of optimising service delivery in a mobile environment. The framework is used as a guideline for identifying different aspects of service delivery in a mobile environment and for providing a path for future research in this field. The focus of the thesis is placed on the service delivery mechanisms that are responsible for optimising the QoS and managing network traffic.

I present a solution for managing traffic through dynamic service localisation according to user mobility and device connectivity. I implement a prototype of the solution in a virtualised environment as a proof of concept and demonstrate the functionality and results gathered from experimentation.

Finally, I present a new approach to modelling network performance by taking into account user mobility. The model considers the overall performance of a persistent connection as the mobile node switches between different networks. Results from the model can be used to determine which networks will negatively affect application performance and what impact they will have for the duration of the user's movement. The proposed model is evaluated using an analytical approach.

Dedicated to my family, for their love and patience…

Acknowledgements

Among family, I would like to thank my parents for their support during these last four years and for helping me with organising my wedding in the final year of the PhD. I would also like to thank my uncle Spiro and his wife Georgia for providing me with a home to stay and for their continuing support for all the years of my studies. I extend my gratitude to my wife Alla and her parents Sergei and Rada for the support and valuable information that they provided and helped me achieve breakthroughs in my research. Their invaluable knowledge on the structure of the Internet and the management of traffic and running costs for network operators has made it possible for me to focus my efforts in the right direction.

I am grateful to Middlesex University for providing me with two scholarships (at MSc and PhD levels) during my academic years and the opportunity it gave me to gain teaching experience and through it, improve my understanding of many network technologies and share this knowledge with future I.T. professionals. Special thanks go to my supervisor, Dr Glenford Mapp for giving me the opportunity to join the university as a researcher and for sharing his knowledge with me from the first day of my studies. Glenford's friendship has been a blessing for me in life inside and outside of academia.

List of Publications

- **Sardis, F.**, Mapp, G., Loo, J., & Aiash, M. 2014. *Dynamic Traffic Management for Interactive Cloud Services*. 7th IEEE/ACM International Conference on Utility and Cloud Computing.

- **Sardis, F.**, Mapp, G., Loo, J.; Aiash, M., 2014. *Dynamic Edge-Caching for Mobile Users: Minimising Inter-AS traffic by Moving Cloud Services and VMs*. Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on, vol., no., pp.144-149.

- **Sardis, F.**, Mapp, G., & Loo, J. (2014). *Cloud-Based Service Delivery Architecture with Service-Populating and Mobility-Aware Mechanisms*. In J. Rodrigues, K. Lin, & J. Lloret (Eds.) Mobile Networks and Cloud Computing Convergence for Progressive Services and Applications. Hershey, pp. 183-199.

- **Sardis, F.**; Mapp, G.; Loo, J.; Aiash, M.; Vinel, A, 2013. *On the Investigation of Cloud-Based Mobile Media Environments With Service-Populating and QoS-Aware Mechanisms*. IEEE Transactions on Multimedia*,* vol.15, no.4, pp.769-777.

- Aiash, M., Mapp, G., Lasebae, A; Loo, J., **Sardis, F.**; Phan, R.C.-W., Augusto, M., Moreira, E., & Vanni, R., 2012. *A survey of potential architectures for communication in heterogeneous networks*. Wireless Telecommunications Symposium (WTS), pp.1-6.

- **Sardis, F.**, Mapp, G., & Loo, J. 2011*. On-Demand Service Delivery for Mobile Networks*. In MOBILITY 2011, The First International Conference on Mobile Services, Resources, and Users, pp. 22-27.

Contents

List of Figures

List of Tables

List of Acronyms

ARP	Address Resolution Protocol
AS	Autonomous System
ASN	Autonomous System Number
CDN	Content Delivery Network
CR/TR- Motion	Checkpoint Recover, Trace Replay
DNS	Domain Name System
ETM	Economic Traffic Management
GUI	Graphical User Interface
IaaS	Infrastructure as a Service
IP	Internet Protocol
ISP	Internet Service Provider
IXP	Internet Exchange Point
LTE	Long-Term Evolution
MAC	Media Access Control
MCC	Mobile Cloud Computing
MEC	Media-Edge Cloud
NDT	Network Dwell Time
NFV	Network Function Virtualisation
PaaS	Platform as a Service
PoP	Point of Presence
QoE	Quality of Experience
QoS	Quality of Service
RDC	Remote Desktop Connection
RDMA	Remote Direct Memory Access
SaaS	Software as a Service
SAN	Storage Area Network
SDN	Software-Defined Networks
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

VHD	Virtual Hard Disk
VM	Virtual Machine
WAN	Wide-Area Network

Chapter 1 Introduction

1.1 Overview

Advances in mobile device technology along with the development of faster wireless connectivity technologies in recent years have given us the ability to access online content and services in a mobile environment. The capabilities of these devices improve with each product generation; however, their small size limits the available on-package resources and ultimately limits the user experience in an era where users demand more multimedia capabilities and multitasking. Advances in wireless network technologies now offer faster network access for these devices and satisfy the need for access to multimedia content and services. Although multimedia content is now available on mobiles, there is still a limit to their capabilities because most of the processing and data storage is done on the device and therefore the limitations of their capabilities still apply when it comes to applications that demand multitasking, processing power and large storage capacity.

The development of Cloud technology has contributed in expanding the capabilities of these devices by offering services and resources over the network. At the moment, the most common example of using the Cloud for expanding the capabilities of mobile devices is that of online storage, and less commonly, the partial processing of information such as for voice recognition. As Cloud technology improves and more sophisticated applications are being developed for the Cloud, we can anticipate a broader use of task offloading from mobile devices to the Cloud and a tighter coupling between the two as the former may eventually transform into a thin-client and rely on Cloud resources for processing and storage. Mobile Cloud Computing (MCC) [14] is the term used to describe this convergence of mobile technology and the Cloud for the purpose of augmenting the capabilities of the former.

To achieve constant access to online resources, mobile devices feature multiple network interfaces of different technologies that increase the chances of acquiring network connectivity. With a broad availability of Wi-Fi, 3G and LTE networks in urban areas, modern devices can connect to the Internet from almost any location. However, it is not yet possible to switch seamlessly between these networks which often results in intermittent connectivity as the device hops from one network to the next. Research in the field 5th Generation (5G) [39] networks is aimed, among other things, at providing reliable and constant connectivity by anticipating the user's moves and proactively configuring network connections so that communication is not lost when the device leaves the coverage area of one network and enters another.

By achieving constant connectivity, it will be possible for mobile devices to rely more on the Cloud for processing and storage and greatly expand their capabilities. However, even with seamless handover technology, each network has different performance characteristic resulting in varying network conditions as the user moves. Furthermore, this raises the problem of managing the additional network traffic generated by these services since task offloading is not subject to caching due to the highly personalised nature of it and therefore it is bound to create large amount of inter-Autonomous System (inter-AS) traffic that congests the Internet on a global scale and ultimately affects performance.

At the moment, MCC offers general purpose services such as voice and image recognition. Examples of it are Apple's Siri, Microsoft's Cortana and City Lens and Google's Voice Recognition for Android. These services do not carry user-specific content and they are easy to replicate across multiple locations to provide the best possible performance for each network. However, as MCC technology advances, more tasks and user-specific content will be offloaded to the Cloud and therefore the mobile device will tend to become a thin-client with most of the processing and storage done in the Cloud. Consequently, it will be inefficient to replicate services that carry user-specific content in multiple locations and therefore network traffic may have to cross multiple networks to reach the client, thus making it subject to performance degradation due to latency. Furthermore, with the service residing in a datacentre away from the user's network, the traffic generated between the service and the client will not be localised and will contribute to the congestion of third party networks.

The convergence of 5G networks and MCC offers the opportunity to provide task offloading services to mobile devices in an environment where connectivity is guaranteed and the capabilities of mobile devices can be reliably augmented without having to rely on mobile hardware for complex tasks. However, there are several challenges that will have to be addressed in order to ensure consistent network performance as per the application’s needs and management of the traffic generated by offloading tasks.

1.2 Challenges

When it comes to managing traffic and improving network performance, the long established method of caching is used. However, when dealing with real-time dynamic content such as a user’s personal service, caching cannot provide the solution since most of the content is generated through user interaction from personal content. For each user, the current set of active applications and data in the memory is unique and therefore we cannot replicate it across many locations. This means that caching using existing methods is not applicable to task offloading. However, Cloud technology has the ability of moving workloads between hosts and datacentres [28, 43] and this capability may be used to dynamically move a user’s service to locations where traffic will be localised and network performance will be improved. Therefore, we can investigate a potential method of dynamically localising services for mobile users based on the network that the mobile device is attached to.

In order to evaluate if a service should be localised, we first need to gather information in terms of network performance and user mobility. From a network’s perspective, moving a service involves transferring a large amount of data between locations and this adds to the traffic congestion and ultimately degrades network performance. Furthermore, if a service is not moved quickly enough, the user may roam to a different network by the time the service transfer completes and thus may create a situation where a service is moving perpetually in an oscillating manner. This would be a disaster scenario as the amount of data transferred over the network would escalate as more services are transferred according to their user’s needs. Hence, the challenge is to construct a mechanism for service migrations that will only move a service when it is deemed necessary and would contribute in the overall reduction of Internet traffic by localising parts of this traffic and ensuring that the user will maintain their connectivity to a particular network for enough time to make up for the extra traffic generated by the migration.

More importantly, network performance is another determining factor to the overall performance of an application when parts of it or as a whole are being offloaded to the

Cloud. The Quality of Service (QoS) [35] of a network is determined by several factors such as packet loss, latency, jitter and bandwidth. As a mobile user moves and the mobile device's network attachment changes, QoS may fluctuate resulting in adverse effects for the service and the user experience. Furthermore, different services may require different levels of QoS from the network and as a result, service localisation should consider these requirements instead of naively trying to localise the service to a location with the best possible QoS. Once again, this is to ensure that services are not in perpetual migration as the user moves from one network to another which would ultimately have the opposite effect.

In order to address the above challenges, we must consider a series of parameters to be used as input for calculating when a service should be migrated and to where. The first parameter is user mobility and it includes the user location, the speed and direction of movement as well as the networks that are available on the user's path. The second parameter is the amount of traffic that the service is generating which is used along with the size of the service to be transferred in order to determine if any traffic savings can be achieved through localisation. To determine this, we have to consider user mobility as an indicator to the duration of a connection to a particular network and therefore how it contributes in calculating the total amount of traffic generated by a service towards a specific network. Finally, we need to consider the QoS of the network and compare it to the QoS requirements of a service to determine if the networks in the user's path satisfy the requirements of the service. In this case we need to model the overall QoS that the service will receive from all the networks along the user's path. By modelling the overall performance of a user's connection along their path, we are able to identify networks that do not comply with QoS requirements and either instruct the mobile device to avoid those or instruct the service to move to a new location where the QoS conditions will improve for the problematic network.

1.3 Research Question

Targeting the challenges presented above, this thesis attempts to answer the following question:

How can we achieve traffic management and enhance network QoS through dynamic service localisation in response to client mobility across heterogeneous networks in the context of Mobile Cloud Computing?

This can be broken down into the following questions:

- ***What are the characteristics of a service delivery framework that can assist in optimising the delivery of Cloud services in a mobile environment?***
- ***What are the traffic management considerations in terms of dynamically localising services according to a user's mobility characteristics?***
- ***How can we develop a QoS that takes into account user mobility and how can it be used to optimise service delivery?***

1.4 Limitations of Scope

As this study mainly delves into the migration decision phase of a service which takes place before a service migration occurs, potential limitations of Cloud technology in migrating services over Wide-Area Networks (WAN) are not considered as they fall outside the scope of this thesis. Instead, it assumes that network performance requirements for service migration over WAN are satisfied and takes into account the throughput of the network in order to determine how quickly a migration can complete.

When a mobile device switches between networks, it typically involves a change in the network address of the device. Achieving seamless connectivity as mobile devices switch networks falls under the research domain of ubiquitous communication. Similarly, moving a service to a different network will also require a reconfiguration of network parameters that are akin to a mobile device switching networks. This thesis assumes that 5G network mechanisms that dynamically and proactively reconfigure the network are in place and uninterrupted connectivity is achieved at both ends of the connection.

Although Cloud technology is used and Cloud Interoperability is presented in the following chapters as an enabling technology for the work proposed in this thesis, the development of actual Cloud Interoperability mechanisms falls outside the scope of this project. A Service Delivery framework is proposed as part of the solution and it is structured in a way that conforms with Cloud's Service-Oriented Architecture, however, it should not be considered an implementation model or as part of a concrete solution to the research question. Instead it is presented as guidance for this thesis and for future work in this topic and in the general context of Service-Oriented Architectures.

1.5 Network Performance in the Context of this Thesis

When it comes to network performance in the context of Cloud applications, one of the most demanding scenarios is Cloud gaming which is a paradigm of centralised processing and user interaction via thin-clients. Researchers in this area focus their efforts on network bandwidth and latency in order to enhance the audio-visual quality and minimise interaction delay [7, 60, and 69]. Other factors that affect network performance and therefore the QoS include jitter and throughput. Table 1.1 presents the performance metrics that synthesise network QoS.

Table 1.1 Network performance synthesis table.

	Network Performance
Bandwidth	The theoretical bitrate capacity of a link
Latency	The time interval between the transmission of a signal from a source and its reception by the destination
Jitter	The variation of latency
Roundtrip Time	The time interval between the transmission of a signal from a host and the reception of a response for that signal from a remote host.

Chen et al. [7] propose a methodology for measuring interaction delay by taking into account the network delay, the processing delay and the playout delay. They define as network delay, the time it takes to deliver a player's command to the server and return an output to the client. Therefore, the network delay is defined as the roundtrip time between server and client. Processing delay is defined as the time it takes for the server to render and output a frame after it has received a command. Hence, the processing delay is a product of the performance of the server and depends on its software and hardware configuration. Finally, the playout delay is defined as the time it takes for the thin client to display a frame on screen after it has received it. In essence, the playout delay is determined by the processing performance of the thin-client. With all other factors constant, these three parameters together define the overall response delay of a

Cloud gaming platform. This generalised approach can also be applied to any Cloud application that uses a thin-client paradigm as the three components are the same regardless of the type of application. Their results show that the OnLive Cloud gaming platform exhibits a lower processing delay for action/shooting games and conclude that this must be due to special configuration in the Cloud to accommodate for fast-paced action games that require a very high level of interaction compared to slower games such as strategy. However, this highlights that although the performance of the Cloud may be configured accordingly, the network delay factor is not something that can be controlled by the user or the Cloud provider and hence it can potentially have a negative effect on the performance without a means for correcting it in real-time. Table 1.2 presents the factors that affect the overall performance of Cloud gaming applications.

Table 1.2 Factors affecting interaction in Cloud gaming.

Cloud Gaming Interaction Delay	
Processing Delay	The time it takes for a Cloud to process a command and produce a corresponding frame
Network Delay	The time taken by the network to deliver player commands to the server and return the corresponding frames to the client
Playout Delay	The time difference between the moment the thin-client receives an encoded frame and the moment it is rendered on screen.

In order to understand how the Quality of Experience (QoE) [34] is affected by the network latency, Wen P. and Hsiao, H. (2014) [70] set up a Cloud gaming test platform at the National Chiao Tung University campus in Taiwan. In their experiments among other things, they artificially limited network bandwidth using a software solution and evaluated the impact on user experience as the roundtrip delay increased. They tested three different games representing action, fighting and shooting genres which all require a quick interaction from the user. They found that for 5Mb/s bandwidth and above the QoE was not affected with all the gamers in their sample reporting equally high levels of satisfaction in terms of interaction and video quality. However, for 2Mb/s and below they found that QoE rapidly declined with their gamer sample reporting very high interaction delay and low video quality which degraded the overall gameplay experience. Consequently, they found that roundtrip latency has a very high correlation to the QoE and that there is a clear bandwidth for each game below which the user experience severely degrades.

Shea et al. (2013) [61] focus their efforts on understanding how OnLive behaves under different bandwidth and latency conditions. In their experiments they found that OnLive implements proprietary mechanisms that shape the internal processing delay of the Cloud to compensate for high network latency. Their findings show that OnLive can compensate at up to 50ms network latency, while trying to keep overall interaction delay below 200ms, however for over 50ms latency, the user experience starts to severely degrade for fast-paced games as the interaction delay goes over 250ms. Similarly, their experiments with network bandwidth show that OnLive will compensate for lower bandwidths by increasing the image compression in order to maintain an acceptable framerate for the user. They compared the Signal-to-Noise Ratio (SNR) of a locally rendered game to the SNR of the same game played on OnLive at different bandwidth values. The highest SNR of 30 was found to be that of the locally rendered game and the SNR from OnLive ranging between 26.70 to 24.41 for bandwidth values between 10Mb/s and 3Mb/s. Their conclusions indicate that although average SNR of 25 is not excellent the image quality is acceptable on mobile devices.

In summary, the main network performance characteristics that affect Cloud services are latency and bandwidth. This thesis does not attempt to define what is "sufficient" performance of the network for a given application but rather attempts to configure Clouds and services in such way that satisfies a given application's demands. Different applications and users can have different needs and consider a different level of performance from the network as sufficient. It is, therefore, up to the users and service providers to define the requirements for their services and based on those parameters; the mechanisms proposed in this thesis will attempt to localise services when the network cannot satisfy these requirements.

1.6 Structure of Thesis

The rest of the thesis is structured as follows: Chapter 2 reviews past attempts at supporting mobility and task offloading in a mobile environment. It continues with the development of Cloud technology and the subsequent convergence between the Cloud and mobile devices. Chapter 3 discusses the evolution of the Internet’s structure and presents new technologies for managing and enhancing the performance of networks. It continues with presenting insights and challenges for the future Internet. Chapter 4 presents a novel service delivery framework centred on 5G networks, Cloud technology and mobile environments. This chapter also sets the focus for the following chapters of the thesis which are about traffic management and QoS enhancement for mobile networks. Chapter 5 presents the mathematical model for managing traffic in a mobile environment using dynamic localisation of services. The chapter continues with a presentation of a prototype implementation using a basic virtualisation platform. Chapter 6 presents and discusses a queueing model which takes into account user mobility and can be used for evaluating the performance of a persistent connection across multiple networks in a 5G environment. Chapter 7 concludes the thesis and presents planned future work. Appendix A contains the script code used in the prototype system as well as screenshots of the script in operation. Finally, Appendix B contains the mathematical solution to a queueing model.

Chapter 2 Task Offloading and Cloud Technology

This chapter presents related work in the context of this thesis. Task offloading for mobile devices in pre-Cloud technology era is covered first, followed by an introduction to Cloud technology and modern techniques for managing tasks in the Cloud. Task offloading for mobile devices using Cloud technology is introduced and the emphasis is placed on the performance of network access technologies.

2.1 Task Offloading in Pre-Cloud Era

This section presents how task offloading was developed in the pre-Cloud era for the purpose of augmenting the capabilities of mobile devices. Some of the more prominent technologies for dynamic task deployment are highlighted.

2.1.1 X Window Teleporting – Early Attempts at Remote Execution

One of the earliest attempts at remotely accessing a desktop environment was the X Window Teleport system developed by Richardson et al. (1994) [55]. The X Window system uses a client-server model to create a Graphical User Interface (GUI) on the screen and accepts keyboard and mouse input from the user. The client and server communicate via network protocols which allow the server and client to operate locally on the same machine or remotely with each residing on a different physical host. The X Server is responsible for receiving commands from the Applications (clients) and rendering the GUI for each running application.

The Teleporting System uses the X Window client-server model to decouple the clients from a particular X Server by introducing a proxy server between them. The function of the proxy server is to receive the rendering commands from the clients and forward them to a user-defined host running X Server, thus giving the user the ability to select in which terminal they want their applications to appear. The initial setup of the Teleporting System allowed the configuration of the proxy via a command line and required from the users to manually enter the target terminal's hostname. In an attempt to automate this task and support user mobility, the hostnames of X Server terminals and their locations within a building were tagged. Infrared transmitters in the form of badges were given to members of staff, each one transmitting their own unique signal which identified the user. Rooms in the building featured networked infrared receivers which identified the user's location and matched it to the hostnames of tagged equipment in that location. This information was passed to the proxy server and the user's session could automatically be teleported to a local terminal.

Unlike traditional Remote Desktop Applications (VNC, Teamviewer, and Microsoft RDC) where the remote host receives a bitmap image of the source's desktop environment and refreshes it in regular intervals or upon user interaction, the Teleporting System completely offloads the GUI processing tasks to the remote host and only transmits rendering commands over the network. This approach is more efficient in terms of processing and network utilisation since the task is distributed and bandwidth-demanding rendered images are not transmitted over the network. Although the Teleporting System had its limitations, it performed its main task of providing remote access and user mobility support successfully.

What Richardson et al. did not realise at the time is that the Teleporting System was also providing task offload functionality by decoupling the user interface rendering from where the applications are actually running. This position is also supported by Chen and Noble (2001) [8] where they argue that Teleporting can be used for centralising the processing and access to the applications can be achieved via stateless mobile thin-clients. Using thin-clients as terminals, the Teleporting System provided a solution for centralising processing resources to be used by demanding applications while giving access to users through any device that supported the X Window system in any location where network connectivity was available. This kind of client/service remote execution in the context of user mobility can be loosely considered as a precursor to Cyber Foraging for mobile clients.

2.1.2 Cyber Foraging – Remote Execution in a Mobile Environment

In the context of task offloading and mobile devices, the main focus is to develop mechanisms and systems that allow mobile devices to expand their inherently limited local resources by accessing nearby devices and acquiring processing capabilities from them. This concept is known as adaptive offloading or Cyber Foraging and it can greatly expand the capabilities of a mobile device to the extent of carrying out tasks that would be otherwise impossible to run on these devices as well as enhance the performance of demanding applications.

In the article "Adaptive Offloading for Pervasive Computing", Gu et al. (2004) [24] argue that the limited resources found in mobile devices often require explicit proprietary programming in order for software to fully utilise their capabilities and achieve maximum performance. The diversity of these devices means that applications have to be adapted to each type of device causing a delay in production and increased development costs. The solution presented assumes that applications can be written in any object-oriented language and the devices running them are able to connect to nearby surrogate nodes to where tasks can be offloaded. One of the requirements is that the mobile node has plentiful wireless bandwidth and has access to surrogates present in the local network.

Gu at al. designed their foraging system so that applications can offload part of their memory requirements to a nearby surrogate in order to avoid crashes due to insufficient memory. The authors experimented with a prototype implementation of their system, using Java programs and found that memory offloading worked but had an impact in performance. In the series of tests conducted, they artificially restricted the amount of memory available to an application running on a PDA. They found that in this scenario the application crashed; however, with an unrestricted amount of memory the application would operate normally. When employing memory offloading to a laptop which served as the surrogate server, the execution time of the application increased, however the offloading engine guaranteed that an application would continue to run instead of crashing when reaching the memory limit. Gu et al. [24] state that this small performance cost is worth the benefit of allowing applications to run normally on a device that would otherwise be unable to support them. This was particularly true at the time the article was written, when most PDAs had approximately 128MB of RAM.

2.1.2.1 Slingshot

A more advanced Cyber Foraging architecture called Slingshot was presented by Su and Flinn (2005) [63]. Slingshot uses service replicas that provide processing resources to mobile applications in order to enhance their performance. Therefore, this architecture is not only foraging for extra memory but also offloads application tasks to nearby surrogates for processing. This is achieved by employing a *"Home Server"* where a first class replica of a service is instantiated for the purpose of persistent storage and task offloading and second class replicas which are temporary replicas running on surrogates local to the user's network and only deal with processing. With mobility in mind, second class replicas only keep a soft state of the application and use the Home Server as persistent storage for creating new replicas when necessary.

In order to support widespread use of surrogates, they have to be easy to manage and require low maintenance. This means that surrogates can be embedded devices and to facilitate this, Slingshot runs each replica in its own self-contained virtual machine. A heavyweight virtualisation platform is used at the surrogate meaning that the operating system running on the surrogate does not pose any restrictions on the offloaded processes. This also means that rebooting a surrogate does not interfere with application behaviour since the surrogate itself only runs a soft state. The only effect to the offloaded applications in the event of surrogate failure would be performance degradation to the level that would have been available had the surrogate never been present.

Slingshot assumes that applications are deterministic, in other words, the result of an execution starting from the same initial state will be the same between different surrogates. Replicas are initiated by checkpointing the first class replica, moving its volatile state to a surrogate and reconstructing the operations that occurred after the checkpoint. It is also assumed that surrogates are connected to the user's local network because Wi-Fi bandwidth is higher and can therefore provide better QoS compared to a backhaul connection. The user's mobile device operates as a proxy between the first class replica and the second class replicas and in some configurations it also serves as storage for the volatile state of the applications and therefore responsible for synchronising a newly instantiated second class replica with updated application context. This is arguably easier and more efficient to achieve over a Wi-Fi link with ample bandwidth than a backhaul link with high latency and less bandwidth.

Slingshot's prototype implementation assumes a single application running at the client and a single surrogate available. The application is divided into a local component which runs on the mobile client and a remote service which is replicated at the Home Server. The resource intensive functionality is ideally allocated to the remote service and the local component contains the user interface. This way, demanding applications can run on clients that would otherwise have insufficient resources for them. The Home Server is assumed to be a machine under the user's administrative control while the surrogates can be administered by third parties. The similarity with the Teleporting system is that the client device only renders the interface while complex computational tasks are carried out remotely.

In the prototype implementation, Su and Flinn experimented using a voice recognition application as an example of a stateless application and a remote desktop as an example of stateful application. They used storage compression (Cox et al. 2002) [13] and content-addressable storage methods (Sapuntzakis et al. 2002 and Tolia et al. 2004) [57, 65] to reduce the amount of physical space required to store volatile and persistent application states. They also used caching at the surrogates in combination with content-addressable storage which reduced the amount of data blocks that needed to be transferred from a Home Server by allowing multiple users of a surrogate to share identical blocks. For example, two users occupying a surrogate and both using a Windows XP VM, will be able to share identical blocks of memory without fetching and keeping separate copies for each client. Regardless, creating a second-class replica is considered a slow process because even after compression, the data to be replicated is in the order of several Megabytes and thus, the transfer time is in the order of several minutes. The results of the experiments showed improvements in the execution times for both applications tested with Slingshot performing up to 2.6 times faster when tasks are offloaded to a nearby surrogate as opposed to remote execution on the Home Server. The greatest performance advantage comes from eliminating communication over the backhaul connection.

With regard to the performance of creating second class replicas, the researchers experimented with storing the service state at the Home Server and at the local client via a Microdrive on the mobile device. For a stateless service, when the service operations are stored on the mobile device the transfer of data to the surrogate for creating a second class replica occurs via the wireless interface rather than the backhaul connection. This results in a much faster instantiation of a replica at the cost of higher processing cost on the mobile device which results in performance degradation in the order of 20% slower response time compared to remote execution. In the stateful service scenario, the performance impact increases to 52% when storing the state on the mobile device. When storing the state on the Home Server, the average response time of the application increased by 20% due to the traffic generated by the VNC connection over the backhaul.

2.1.2.2 Dynamic Service Deployment

In a similar effort to offload processing from mobile devices by means of foraging, Verbelen et al. (2011) [67] presented a prototype model for dynamic deployment and quality adaptation for mobile augmented reality applications. Their approach requires mobile applications to be written in bundles which are then used to adapt the application to the resources available on the mobile devices and on surrogates near the device. The advantage of their approach is that it allows for more fine-grained control of the performance of an application when it is being starved of resources. As an example, they implemented an augmented reality application which uses the mobile phone's camera to identify items via their barcode or their label. When the user is multitasking on the phone, the application can call the different bundles that comprise it and adjust its performance to enhance the user's QoE. Furthermore, if a surrogate is detected on the network, it can offload some of these bundles and further improve the performance of the application and hence the user experience. Verbelen et al. experimented with different configurations by altering the amount of resources available to the device through multitasking and by adding surrogates to the network. They also experimented by altering the bandwidth available between the device and the surrogate. In their conclusion, the framework enhances the application's performance and the user's perceived QoE with only momentary degradation of the experience when the application is switching between bundles and is being redeployed. They argue that the redeployment phase can be further improved by employing bundle caching on the surrogate.

2.1.2.3 Cyber Foraging Summary and Challenges

In summary, the concept of foraging offers a dynamic method of allocating resources to devices that inherently lack them. Performance improvements can be achieved under certain conditions but there are also several drawbacks. Security is one such drawback considering that users will be temporarily storing and processing their data on a public machine administered by a third party over which the user has no control. Security in virtualised environments is a problem directly affecting foraging but it is not entirely in the field of foraging to solve. The other notable drawback of foraging is the complexity of deploying parts of an application to a surrogate when dealing with stateful applications that need to be synchronised across multiple surrogates. The backhaul connection may pose a bottleneck when synchronising states or transferring data and when the synchronisation is driven by the mobile client, the performance penalty may lead to degradation of the user's experience. Perhaps the most important drawback when it comes to foraging is that although it is made with mobile devices in mind, it is not centred on user mobility in an automated and dynamic manner. Slingshot for example, employs a user-initiated instantiation of replicas which implies that the user is responsible for evaluating whether or not offloading is required. It also expects the user to assess for how long they are bound to stay at a location and connected to the native hotspot and at the same time the user is required to know the signal strength of the hotspot and if it provides adequate bandwidth (as per the application's needs) for offloading. Additionally, the required complexity for creating applications that are compatible with offloading parts of their operations may create additional development costs which negates the initial argument of reducing costs by making applications agnostic to their environment and running them within a Virtual Machine.

Another notable concern with foraging is the amount of traffic it generates on networks. When dealing with small-footprint applications, replicating parts of their functionality over the network may not generate a large amount of traffic. However, as technology progresses and users are becoming accustomed to complex applications on their mobile devices, this process may consume a lot of bandwidth for an extended amount of time. For example, when playing a game, image rendering is a complex task that needs to be remotely executed. This means that the entire application may have to run on a surrogate and transmit rendered images to the client, thus heavily congesting the network with a traffic flow that has high bandwidth and low latency requirements in order to sustain a good QoE.

Furthermore, if we assume that surrogates are native to public hotspots used by many people concurrently, the question of scalability arises. How powerful will surrogates need to be to support this functionality for tens of people? How much bandwidth will the hotspot need in order to accommodate each user's needs? Will the backhaul connection of each hotspot have sufficient throughput for multiple concurrent synchronisations without affecting performance?

With regards to the amount of available resources and the scalability of deployment, the development of Cloud technology has provided answers to many of the problems faced by Cyber Foraging. The next section presents Cloud Computing technology and some of its latest development.

2.2 Cloud Computing Technology

Cloud technology has changed the landscape of remote task execution by offering centralised computing and storage resources to clients. The Cloud has now become and established paradigm for providing services and content on the Internet. This section explores Cloud computing and presents some of the latest developments in the technology.

According to the European Telecommunications Standards Institute (NIST): *"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models."* (NIST, 2011)

According to the NIST, the essential five characteristics of a Cloud are described below:

On-Demand Self Service: *Cloud users can unilaterally provision computing capabilities as needed without having to rely on a third party (technical department) to do it. In other words, Cloud users are presented with a simple interface which allows them to choose to which services of the Cloud they wish to subscribe. It also allows them to deploy their own services and applications on top of Cloud services.*

Broad Network Access: *Cloud capabilities are made available over the network in a secure and reliable manner using standard mechanisms (e.g., HTML 5) that can be used by heterogeneous client platforms (e.g., workstations, tablets, mobile phones).*

Resource Pooling: *A Cloud has a pool of resources (e.g., storage, processing, network bandwidth, and memory) which is used to serve multiple clients by applying a multi-tenancy model. The customers are not aware of the exact location of their resources within the Cloud but they may be able to specify a location at a higher level of abstraction such as a geographical region or datacentre.*

Rapid Elasticity: *Cloud resources (physical and virtual) can be dynamically assigned and reassigned to various services based on client demand. As the demand for a particular service increases, more resources are allocated to it dynamically. When the demand decreases, the excess resources are returned to the pool so that other applications can use them.*

Measured Service: *Cloud systems must be able to monitor, control and report the amount of resources used by each tenant of the Cloud. The Cloud provider is able to measure the cost of resources used by each tenant and charge accordingly.*

Cloud technology can offer a wide range of services at different levels of abstraction. They are most commonly classified by the NIST in the following three categories:

Infrastructure as a Service (IaaS): *This is the most basic services provided by a Cloud. The consumer is presented with a virtual infrastructure which includes processing, network and storage resources. The infrastructure can be used to implement an entire virtualised network with hosts, operating systems and applications. The consumer does not control or manage the underlying Cloud infrastructure.*

Platform as a Service (PaaS): *Going one level up from IaaS, in PaaS, the consumer is presented with a complete platform which can be used for deploying applications or creating applications using various programming languages, libraries, tools and services. The consumer does not manage the infrastructure of the platform or anything underlying.*

Software as a Service (SaaS): *This service type gives consumers the ability to use the provider's applications (e.g., web-based email) through a wide range of client* *devices such as mobile phones, tablets and laptops. The consumer does not manage any of the underlying infrastructure and the applications are typically presented through a web browser or a program interface.*

Although Cloud computing may sound as a revolutionary approach to service provisioning, in reality, the types of services it offers have existed for many years before Cloud became mainstream. Even the method of delivery for some of these services has not changed. What Cloud technology revolutionised was the platform on which services are running. It allows us to consolidate hardware resources and efficiently allocate them to different services and clients based on availability and demand. In Fig. 2.1, below we present the Cloud services along with their precursors as defined by Rhoton, (2009) [54].

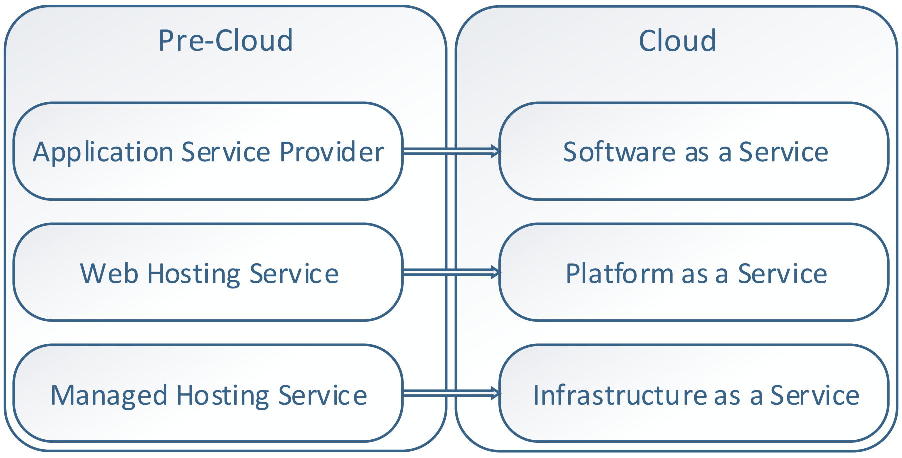


Figure 2.1 Cloud services and their legacy equivalents.

Clouds can be deployed in four main ways as defined by the NIST. Each deployment type has different advantages and applications depending on the requirements that an organisation:

Private Cloud: *A Private Cloud is entirely dedicated to a single organisation. The Cloud infrastructure may be located on or off premises and may be managed by the organisation, a third party or both. Typically, with on-premises deployment, the organisation has full control of the datacentre's infrastructure and networks. Off-premises private Clouds usually take advantage of existing facilities and expertise of a third-party which is tasked with hosting and maintaining the datacentre. The advantage of private Cloud is that the organisation can keep a tight control over it in terms of design,* *implementation, maintenance and operation. The disadvantage is the high cost involved in designing, implementing and running it.*

Public Cloud: *A public Cloud is owned, managed and operated by a business, academic or government organization, or some combination of them. It exists in the premises of the organisation and the infrastructure is typically used to sell various services to the general public.*

Community Cloud: *A community Cloud is shared by many organisations that share common concerns such as performance and security. Its infrastructure can be managed internally by one or many of the organisations in the community, or by a third party, or by some combination of them. It may exist on or off premises. In essence, it is a Private Cloud implementation that is shared by a community of organisations which share some common goals or requirements and wish to have a tighter control over the infrastructure than a Public Cloud would afford.*

Hybrid Cloud: *A Hybrid Cloud implementation combines two distinct Cloud infrastructures, a private and a public, which co-operate and share resources by means of standardised or proprietary technology that enables data and application portability. In this deployment model, a Private Cloud can be offloading certain tasks to a Public Cloud for load balancing or other purposes.*

2.2.1 Cloud Infrastructure

In the previous section we explained the basic characteristics of a Cloud, what types of services it can offer and its main deployment configurations. In this section we will look at how a Cloud operates and how it offers services. We will start by looking at a basic Cloud network topology as shown in Fig. 2.2.

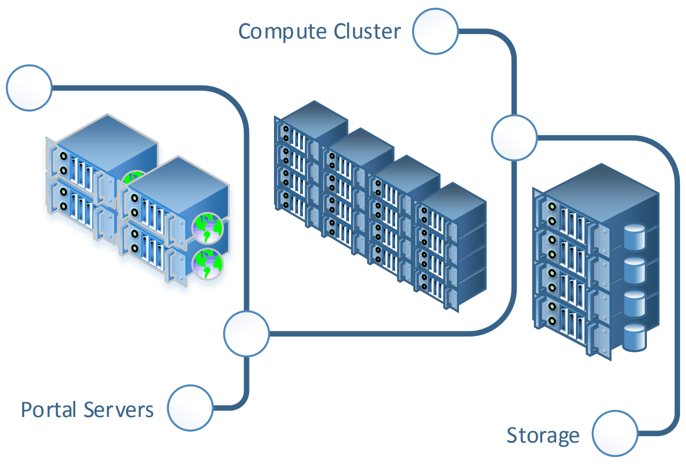


Figure 2.2 Cloud network topology.

A Cloud is by itself a network of networks with different servers within it running different services. Part of the Cloud infrastructure is reserved for storage, typically in the form of a Storage Area Network (SAN). The SAN on its own is an independent network with dedicated storage devices and front-end devices that interface with the rest of the Cloud's network. Long-term storage for clients and for Cloud operations is consolidated within this sub-network. Virtual Machine configurations, Virtual Disks, User Data and Boot Disk Images for the Compute Cluster are stored here. The SAN presents itself as multiple block-level devices. If we take iSCSI as an example, then a SAN can present multiple iSCSI targets that are accessed by the Compute Cluster. Since the iSCSI targets are block-level devices, they appear to the operating system as a hard disk which can be formatted and used in any way. Boot Disk Images used for Pre-Execution Environment (PXE), can also be stored in the SAN and used by the Compute Cluster to deploy a Bare-Metal Hypervisor to new nodes. This allows for fast deployment of new Compute Nodes which in turn-expands the capabilities of a Cloud. Another advantage of having a dedicated storage network is that all the compute resources of the Cloud can be reserved for the clients, while dedicated hardware carries out tasks such as data backup, archival and retrieval, integrity and parity checks, data migration, caching and maintenance.

The Compute Cluster gives a Cloud its processing power. The server nodes residing in this portion of the network are typically configured to boot into a Bare-Metal Hypervisor via PXE. They are all interconnected and managed by other nodes within the cluster that are tasked with distributing workloads and allocating resources. Many of the servers and services that manage the resources of the cluster are themselves virtualised. The Compute Cluster is configured with machines with plenty of memory and processing power in order to handle client tasks. Long-term storage of data is achieved via connections to the SAN. This pool of processing resources can be used to run independent Virtual Machines or virtualised networks of machines. High availability is achieved through failover clustering which constantly monitors nodes and moves services or virtual machines among them when needed.

Finally, the Portal Servers provide the front-end of the Cloud where clients connect and access services. Everything behind the portal is abstracted and the client is presented with only the information that is relevant to the services he is requesting. Clients are given the ability to subscribe to different services, change parameters of their services, calculate the cost of the resources they are consuming and also access their own personalised environment for developing their own platforms or services. All this information and functionality is primarily handled within the back-end of the Cloud which manages the Compute Cluster. This information is accessed by the front-end and presented to the user in a simple interface that can be accessed by a web browser. The front-end can also pass instructions to the back-end for configuring new client tasks. For example, a client can access a VM through the portal and they can gain access to the operating system installed on the VM, but they can also configure how much memory, storage and how many processors they want their VM to have. They can also install a different operating system or new software within the VM. The most common example on the Internet is the various Cloud Storage services such as Google Drive and SkyDrive where the user is presented with a graphical representation of their files via their web browser. Files can be downloaded or uploaded, archived or deleted and they can also be made available to the general public via special links.

As explained above, a Cloud is ultimately a network comprised of smaller, purpose-built networks. Nodes within the networks are connected in a loosely coupled fashion using high bandwidth interconnects that form the fabric of the Cloud. Similarly, those networks are also interconnected to each other with high bandwidth links. As a result, we will see in the next section that the performance of the fabric plays a big role in the overall performance of the Cloud (CISCO, 2014) [10].

2.2.2 Virtual Machine Migration

Moving a Virtual Machine involves transferring one or more components from one physical location to another. This can be done while the VM is not running or while the VM is live. A live migration of a VM is achieved with minimal or no interruption to its service via handover mechanisms that we will discuss in this section. There are two main types of migration that are of interest in the context of this thesis. A "full" migration of a VM transfers the Virtual Hard Disk (VHD) and the VM to a different host. The process is typically started by making a copy of the VHD to the new location followed by a transfer of the VM's memory contents and the VM configuration and finally a network-level handover, if the VM is using a network. A "light" migration transfers only the VM's memory contents to a new host and performs a network-level handover afterwards.

For a VM running within a Cloud with its VHD stored in a SAN, we only need to perform "light" migrations for load balancing or failover scenarios. For a VM running on a simpler setup of virtualised hosts, without a SAN or a centralised storage solution, a "full" migration is more common. The difference between the two types of migration is the time it takes to complete the process. A "full" migration takes a longer time to complete due to the bulk of data of the VHD that has to be transferred. Additionally, the storage system often poses a bottleneck in the operation because its throughput is slower than the network throughput of a Cloud's fabric. In contrast, a "light" migration is purely a memory copy operation and is encountering a bottleneck at the network's throughput which is many orders of magnitude slower than the memory subsystem of a server.

2.2.2.1 Live VM Migration Process

In detail, a "light" migration has six phases (Microsoft, 2012 and Hwang, 2012) [48, 30]. In the first phase, the source host sets up a connection to the destination host and transfers the VM configuration data. At this point, a basic VM is set up at the destination using the configuration data and memory is allocated to it.

In the second phase, the entire memory (working set) of the migrating VM is copied over the network from the source to the destination. As the working set is being migrated, the host monitors which pages were modified since the beginning of the copy operation. Pages that have been modified are added to a list of pages that need to be recopied. This phase is also known as the pre-copy phase.

In the third phase, the hypervisor synchronises the modified pages between the two hosts by looking up the list of modified pages and transmitting the changes. As entries in this list are being transferred, other pages can be modified. It is therefore important to have high network bandwidth between the source and destination hosts such that the page transfer rate exceeds the page modification rate. This process is repeated multiple times until there are no pages left on the list.

In the fourth phase of the migration, any virtual hard disks or other storage associated with the VM have their control transferred to the new instance. At this point, the destination host has an up-to-date working set for the new VM and access to the storage. So in the fifth phase, the new instance of the VM is resumed and brought online.

In the sixth stage, the destination host sends a message to the network switch and updates the MAC address for the VM in the switch's Address Resolution Protocol (ARP) table so that traffic can be forwarded to the correct switch port. The flowchart in Fig.2.3 shows the migration process.

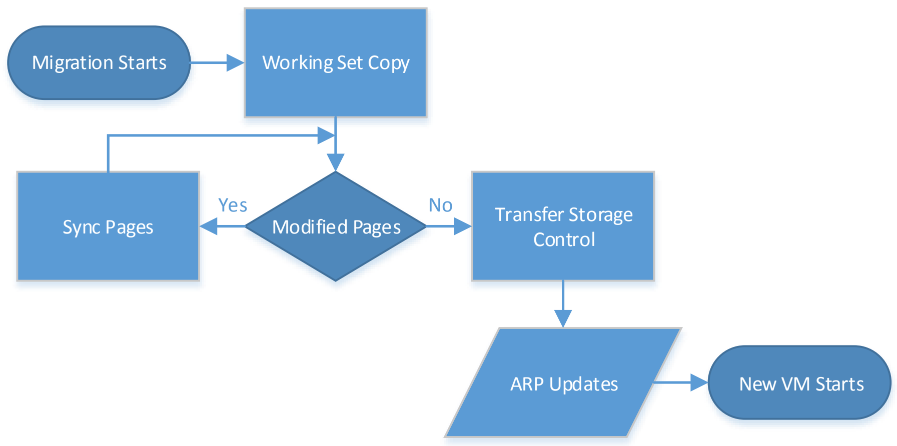


Figure 2.3 Migration process flow diagram.

In the event of a "full" migration, where there is no central infrastructure and storage is not shared, the process of moving the storage occurs before the "light migration" phases begin. While storage is being copied, any read/write operations are forwarded to the source VHD. Following this stage, read/write operations are mirrored on source and destination while any last changes are being synchronised. At this point, the process of "light" migration starts as described above. Once it completes, the source VHD is deleted.

In summary, VM migration is achieved by first copying the working set of a VM in a process that is called pre-copy. Memory pages that were modified during the transfer are added to a list and are iteratively copied. The last set of pages is copied after freezing the VM while at the same time; control of the storage is given to the new VM. The new VM is then brought online and the network is reconfigured to forward the active connections to the correct port on the switch.

2.2.2.2 Migration Performance Analysis and Enhancements

Typically migration performance is measured in terms of migration time and downtime. The migration time is the time it takes for all the phases of the process to complete. It depends on the size of the working data plus the size of the storage data in the event of a "full" migration. It also depends on the performance of the network that connects the source and destination hosts as well as the performance of the hardware of the hosts. The downtime occurs while the last modified pages are synchronised (while the source VM is paused) and the new instance is coming online and updates the ARP tables of the network. To achieve optimal migration performance, these two metrics need to be minimised (Ibrahim et al., 2011) [31].

From the above, it is apparent that "full" migration should be avoided in order to minimise total migration time. Therefore, in an ideal scenario, migrations will occur between hosts that share some kind of storage infrastructure. It is also obvious that in order to complete the migration, the rate of pages being modified at the source should be smaller than the rate of pages being transferred to the destination. Another way to present it is that with each repetition of the copy operation, the list of modified pages should become smaller. Performance of the VM is also impacted during the pre-copy operation while pages are being synchronised across the physical hosts. This is due to copy overheads as the physical host has to dedicate resources in the operation, thus depriving the VM of those resources. An adaptive rate limiting approach can be employed in order to limit the amount of resources dedicated to the migration but as a result the migration time is increased (Hwang, 2012) [30]. Moreover, prolonged migration time means that pages can be modified multiple times during the process without converging to a small writable working set. Therefore, the maximum number of iterations has to be defined in the rate limiting approach, in order to guarantee that the migration will complete without entering a contention loop where pages are modified faster than they can be copied to the destination.

Another approach to improving the performance of live migrations is the Checkpoint Recovery, Trace Replay (CR/TR-Motion) approach. Proposed by Liu et al. (2009) [40], this approach is aimed at improving the migration time, reducing the downtime and minimising the network traffic. It is achieved by taking a "snapshot" or "checkpoint" of the VM's working set. At this point the host monitors the VM's execution and records events in a log file. The original VM keeps operating normally while the snapshot image is transferred to the destination host. Log files are then transmitted from the source host to the destination. These log files are then used to replicate the changes to the VM's working set on the destination host. Therefore, the approach of CR/TR-Motion is to describe the changes that are occurring instead of transmitting the modified pages. Eventually, the log size becomes small enough to warrant a quick stop-and-copy operation. The downside of this approach is the same as the pre-copy approach in that the log replay speed on the destination host must be faster than the log generation on the source. Otherwise the destination system would end up chasing the source's state indefinitely. Because log files typically contain descriptions of the changes, they create less traffic than moving modified pages. For example, a log file less than 1KB may describe changes made to a number of pages over 1MB. Therefore, this approach is successful at reducing the network traffic generated by a migration.

Another method of reducing the downtime and migration time is to compress the working set before copying it to the destination. This approach makes use of free CPU resources on the source and destination hosts. The compression/decompression of pages is not computationally intensive and as a result the migration time is reduced and the network traffic is also minimised (Finn, 2013) [20]. Finally, Remote Direct Memory Access (RDMA) can be used with certain types of hypervisors bringing further benefits to the network throughput of a migration. Xen and Hyper-V support VM migration using RDMA which completely bypasses the TCP/IP stack processing overheads. Although this approach requires specialised Network Interfaces on the hosts, it is capable of transferring the working set of a VM without involving CPU, caches and context switches. The migration occurs directly from the RAM of the source host to the RAM of the destination with only bottleneck being the network bandwidth between them. Multiple network interfaces can be teamed together to achieve very high transfer rates with minimum CPU utilisation on the hosts (Finn, 2013) [20].

2.2.3 Media-Edge Cloud

Zhu et al., (2011) [72] proposed Media-Edge Cloud (MEC) as a platform focusing on localising certain services within a Cloud in such manner that enhances Cloud operations and the QoS. MEC is an attempt at more efficient utilisation of Cloud resources that will give service providers the best performance possible without requiring costly upgrades on their networks. With MEC, QoS-sensitive services are located as close to the Cloud's edge as possible. This means that services such as video streaming are running on nodes closer to the Cloud's client-facing front-end and thus are fewer hops away from the user. The logic behind this approach is that placing these services close to the Cloud's Public Portal, reduces the amount of traffic within the Cloud's fabric. Routers that are deep into the Cloud's Compute Cluster are decongested because the streaming traffic can be passed on directly from the service nodes to the routers on the Portal network. This in turn enhances the QoS within the Cloud by releasing network resources to be used for other purposes. The users can enjoy better interaction with these services and a better QoE due to the decreased latency as a result of the smaller network distance.

Hobfeld et al. (2012) [27] presented a classification of Cloud services based on their level of interaction and media content in an attempt to better understand how these applications are better located within a Cloud in the context of MEC. Services such as games and operating systems are at the top of complexity and interactivity while applications such as on-demand video streaming are at a lower place. A chart with some sample services classified according to this scheme is shown in Figure 2.4Fig. 2.4. Based on this classification, Hobfeld et al. [27] identify the QoE problems in a Cloud service as artifacts introduced by the network distance, resource management due to collocation, geographical distribution of the user base and the number of parties involved in providing the service.

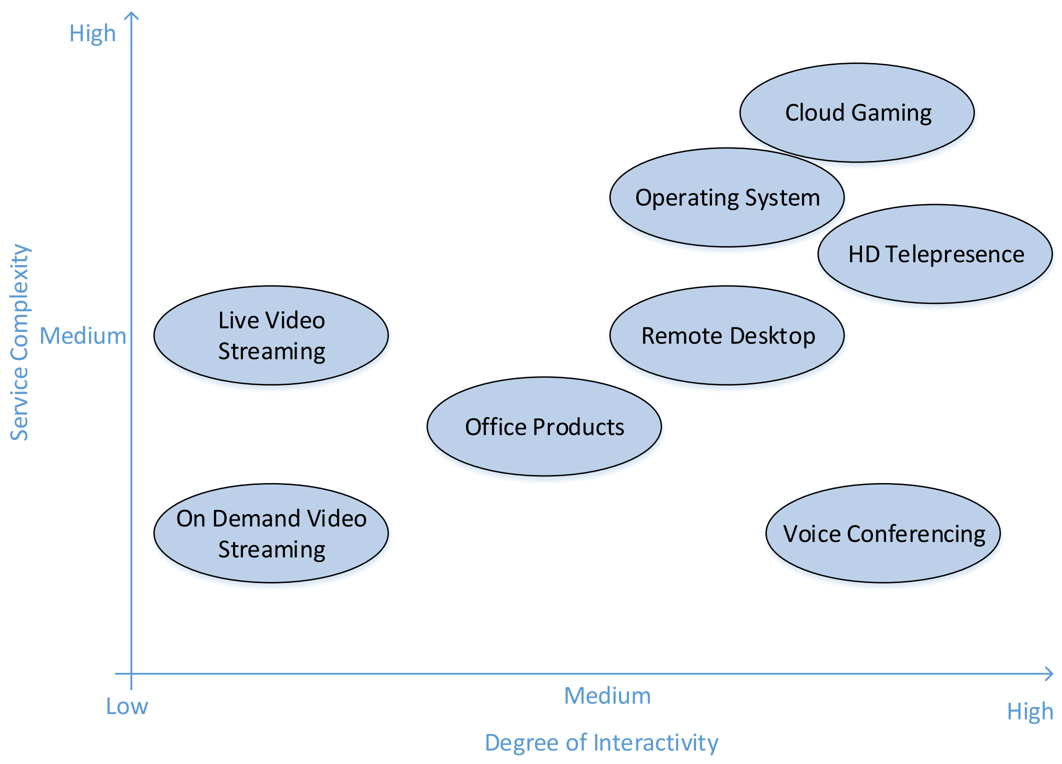


Figure 2.4 Classifications of services according to interactivity and complexity.

A good example of the complexity of QoS challenges in a Cloud environment is Cloud gaming. There are two types of Cloud gaming depending on where the actual processing of the game is being done. In the example of fat-client Cloud gaming, the game runs on the client's device while the Cloud distributes game content as needed (Alexander, 2012) [2]. Textures, level architecture, media files and other game content are stored on the Cloud and the user's client requests this content as and when needed in order to minimise the footprint of the application on the client's machine. On the other hand, thin-client gaming runs the entire application on the Cloud and the user's device receives the rendered image of the game (Alexander, 2012) [3]. User input is transmitted to the Cloud in order to interact with the game which implies that a very good connection to the Cloud is necessary. One thin-client gaming example is OnLive and it is an example of a service that could benefit from the MEC architecture. The various constituent game services can be distributed in the Cloud with the rendering and game logic running closest to the edge of the Cloud to improve QoE while storage may reside deeper in the infrastructure.

However it is worth noting that MEC assumes services to belong to a single class. It does not consider dynamic deployment of services based on their current level of interaction.

Service classification by itself is a static way of dividing services that may not perform well under some scenarios. For example, a video game can be highly interactive (action game) but it may also be more passive (puzzle games, turn-based games). Under the MEC classification, putting a turn-based game on the edge of a Cloud may not be the most efficient use of resources. A more dynamic solution that monitors the level of interaction of a service in real-time is needed.

2.2.4 Cloud Interoperability

The concept of Cloud interoperability revolves around standardising certain aspects of Cloud computing with the aim of achieving portability of workloads and sharing of resources between heterogeneous Clouds. Each Cloud technology has its own specific mechanisms, processes, formats and APIs to achieve various functions within the Cloud. This proprietary approach means that a client workload such as a VM, that is configured to run on one Cloud, is not portable to a different Cloud. This problem has hurt the initial adoption of Cloud technology because many corporations did not want to get locked-in to a specific Cloud technology vendor. Moreover, these differences between Clouds meant that there can be no co-operation between two heterogeneous Clouds in terms of sharing resources and offloading tasks to each other if needed. As a result, hybrid Cloud implementations are forcing the private Cloud deployment to be of the same technology as the public Cloud which in turn results in less flexibility for a corporation or individual.

IEEE Standards Association has two projects on the subject of Cloud interoperability and technology standards. Project 2301 is tasked with advising Cloud ecosystem participants of standards-based practices and choices in terms of application, management and portability interfaces, file formats and operation conventions. Project 2302 is developing the Standard for InterCloud Interoperability and Federation. It defines the topology, functions and governance for Cloud interoperability and federation which includes topological elements (e.g., roots, exchanges), functional elements (e.g., name spaces, trust infrastructure) and governance elements (e.g., registration, auditing). The standard does not address IntraCloud operations and proprietary hybrid Cloud implementations.

ETSI has started a Cloud Standards Coordination initiative which has produced a report aimed at Cloud providers and customers alike, as well as administrators and governmental authorities. It clearly defines the roles and use cases for a Cloud and also addresses the Interoperability perspective in terms of avoiding vendor lock-in by means of data portability, standards compliance and common interfaces (Darmois, 2013) [14]. Proprietary Cloud technology vendors such as Microsoft are also creating Interoperability initiatives addressing issues such as data portability, standards, ease of migration and informing Cloud application developers on the best practices for creating platform-agnostic solutions (Microsoft, 2010) [47].

In October 2013, IEEE launched the InterCloud Testbed project (IEEE, 2013) [32, 33] and announced the founding members which include academic and industry researchers, Cloud companies and service providers. The aim of this project is to bring together organisations that have an interest in Cloud Interoperability and develop the technology necessary to build a testbed platform using technical standards created by P2302 [68]. The architecture of the testbed is analogous to that of the Internet, with private Clouds playing to role of intranets while public Clouds play the role of Internet Service Providers (ISP). The aim is to provide additional resources to private Clouds from heterogeneous public Clouds, making this testbed the prototype of a global scale hybrid Cloud implementation.

2.3 Task Offloading in Cloud Computing Era

This section focuses on the development of Mobile Cloud Computing technology as the foundation for modern era task offloading in mobile devices. The definition of Mobile Cloud Computing is given below, along with some of the more popular and commercially successful technologies for MCC deployment.

2.3.1 Mobile Cloud Computing

According to Dinh et al. (2011) [15] The Mobile Cloud Computing Forum, defines MCC as:

"Mobile Cloud Computing at its simplest refers to an infrastructure where both the data storage and the data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and mobile computing to not just Smart Phone users but a much broader range of mobile subscribers."

In essence, MCC makes use of the Cloud as a means of offloading complex computing tasks from mobile devices to the Cloud for the purpose of augmenting the capabilities of these devices. At the NASA IT Summit, Warner and Karman (2010) [69] went one step further by claiming that when robust connectivity is available on the mobile devices, it is possible to convert them into thin-clients and carry out all the processing in the Cloud. Thus the mobile device itself is only used for presentation and caching purposes.

It becomes apparent that if the mobile device becomes a thin-client and all the user's applications are running on the Cloud, the performance of networks is a determining factor to the overall user experience. Huang (2011) [28] argues the need for a geographically-based distribution of MCC datacentres, such that minimises the distance between services and clients as much as possible to eliminate performance degradation from long network paths. However, Bahl et al. (2012) [4], argue that distributing MCC services to the closest datacentre to the user does not guarantee a good performance even when a modern network technology such as LTE is used. They highlight the need for a middle tier where computing resources will be made available closer to the user than the nearest MCC datacentre and therefore eliminate long connection paths. This bears similarities with Cyber Foraging with the distinction that Public Clouds serve as Home Locations while the role of surrogates is played by smaller Clouds located closer to the user or even within the user's network.

Satyanarayanan et al. (2009) [59] propose the use of Cloudlets as a middle tier in order to enhance the performance of Cloud applications by bringing the resources closer to the user's network. The proposed solution uses the Cloud as a back-end and uses Cloudlets for applications that are sensitive to network performance. When a Cloudlet is not available on the user's network, the mobile device falls back to using the Cloud for offloading. When the user changes networks and a Cloudlet is discovered, tasks are copied over to the Cloudlet using migration techniques in order to improve network performance. Thus, the idea of Cloudlets also bears has many similarities with Cyber Foraging presented in the previous section. They both focus on augmenting the capabilities of mobile devices through task offloading. They both attempt to improve performance by locating the Cloudlets or Surrogates as close as possible to the mobile device. Finally, both ideas feature task portability in real-time. However, their main difference is that Cyber Foraging still relies on the mobile device for most computing tasks while MCC and Cloudlets assume a mobile thin-client and carry out all the processing and storage on the Cloud. The main drawback of Cyber Foraging is that it applied to scenarios of limited mobility, as explained previously. Cloudlets may also have the same drawback since it is implied by Satyanarayanan et al. [59] that a very fine-grained approach to Cloudlet localisation is required, such that individual LANs will feature a Cloudlet for serving their clients. It is unlikely that Cloudlets will be deployed in most LANs in the future and therefore, most of the offloading will be performed in the Cloud. Furthermore, in scenarios of highly mobile clients, there will be a large amount of data transferred on the network as services are constantly being replicated between the Cloud and Cloudlets every time the user is changing networks. Ideally, we need a less fine-grained approach, such that Cloud datacentres can be localised through the use of peering with different networks and therefore provide good service to their peers. Alternatively, we can consider an approach where large scale networks such as Internet Service Providers have smaller datacentres within their networks for the purposes of MCC. We are therefore looking at a form of edge-caching such as employed by Content Delivery Networks (CDN) but applied to MCC datacentres. Instead of replicating content, in this case services are being migrated depending on the network that the user is connected to, thus minimising the network distance between the mobile device and the Cloud.

Bahl et al. (2012) [4] present a set of requirements for moving MCC technology forward. Their position is that Cloudlets are an integral part of MCC as they can help in localising services and enhance the performance of applications. In their agenda, they outline two possible deployment settings for MCC. The first setting involves mobile network operators deploying Cloudlets within their infrastructure and offering their capabilities as premium service for subscribers. The second option involves the Cloud providers forming co-location agreements with mobile network operators for the purpose of providing Cloudlets within the infrastructure of wireless networks. In both settings, the outcome is the same and it involves the use of a public Cloud serving as the back-end for services while Cloudlets deployed within wireless networks act as a middle tier to improve the performance. The set of requirements as outlined by Bahl et al. [4] is as follows: The network will have to support high bandwidth and low latency to the regional datacentres of public Clouds to enable the fast migration of services between Cloudlets and Clouds. The Cloudlets and the public Cloud will have to support the fast migration of services via mechanisms that manage the available resources. Furthermore, the Cloudlets and public Clouds should have a common computing platform and also be aware of service dependencies so that when a service is moved, the relevant and required services will also move along with it. Finally, the Cloud in addition to running services, should also serve as a loosely synchronised persistent storage for services running on Cloudlets. In either of these scenarios, we can see that Cloud Interoperability will negate the need for a common Cloud platform for Cloudlets and Public Clouds, thus enabling heterogeneous Clouds to be deployed by providers and co-operate seamlessly for the provision of services to mobile clients.

2.3.2 Supporting the Case for Mobile Thin-Clients in MCC

Abolfazli et al. (2013) [1], divide MCC into two different approaches. The first approach uses fixed distant Clouds for providing services to mobile devices. The second approach uses nearby Clouds for the same purpose. Cloudlets fall in the second category as they use proximate Clouds to deliver services and Public Clouds to store persistent data. However, the most complete offloading solution that is currently available falls in the first category and is offered by OnLive. OnLive uses distant fixed Clouds to offload all the processing from client devices, thus making it possible to run applications in a completely virtualised environment. Rendering is also performed on the Cloud and the client device acts only as a thin-client to display content and transmit user input. OnLive presents a solution to MCC that removes the complexity of dividing computational tasks between the client device and the Cloud and thus simplifies the implementation model. As pointed out by Abolfazli et al., [1] this concept of MCC enhances the visualisation capability of client devices, promotes cross-platform deployment of applications by negating the impact of platform and hardware heterogeneity and thus facilitates the universal access of a computing platform through various devices such as Smart Phones, laptops and tablets. However, this approach is particularly sensitive to network QoS parameters and in order to maintain a high level of responsiveness in interactive multimedia applications, a low latency and high bandwidth is a requirement.

As highlighted by Abolfazli et al., [1] one of the main challenges for MCC lies in achieving a small Round-Trip latency in order to guarantee good applications performance and high level of responsiveness. Another challenge to consider is the management of traffic generated by MCC and more specifically the traffic cost and congestion management.

However, the benefits of using thin-clients as terminals far outweigh these two challenges as a lot of the complexities in developing frameworks for partitioning processing tasks and applying partial offloading solutions are negated. Therefore, no special programming techniques need to be employed when developing applications and there is no need for creating task-offload engines that distribute parts of applications between the mobile device and the Cloud.

For the reasons outlined above, this thesis supports the idea of MCC and thin clients as the best solution for consolidating processing in the Cloud, developing platform-agnostic applications and simplifying the development and deployment of such applications. However, network performance in this case has a more prominent role to the user experience and therefore the proximate Cloud approach is preferred as a solution for localising traffic and enhancing the QoE.

2.3.3 Cloud Gaming

One of the best ways to test the viability of using thin-clients to display content that is rendered in the Cloud is to examine popular Cloud gaming applications. First popularised by OnLive and Gaikai, the idea behind Cloud gaming is to put game logic and rendering on the Cloud and display the rendered images to a thin client. Thus, gamers need not worry about having the latest hardware capable of playing modern games and also gain the ability of accessing their personal game data from any client without having to download the game. For game developers, this means that games can be made for a Cloud platform without spending development time for optimising them for different platforms and hardware specifications. Before we introduce mobile thin-clients to the idea of Cloud gaming, we need to be aware of parameters that affect the user's QoE.

One of the first things to understand about Cloud gaming is that all user interaction occurs over the network and as a result, network latency needs to be taken into account as well as the processing and rendering latency and finally the image transmission latency. When dealing with local rendering, even when considering an online multiplayer game, interaction delay is not immediately apparent to the gamer because of the low latency between issuing commands and the game engine rendering frames. Even in the case of an online multiplayer game, as long as the game is rendered locally, it is only the server and other players that are experiencing interaction delay with each other. For each individual player, their own actions occur instantly on their screen. However, with Cloud gaming, a player's own actions occur after a delay. This means that care has to be taken in minimising this latency, either be reducing the network latency or rendering latency or both.

Clinch et al. (2012) [11] investigated in a Cloudlet platform, how close to the user, the service should be located in order to provide a good QoE. They set up a test platform with Cloudlets distributed in the user's subnet in the UK and in remote networks within the EU and the USA. Amazon's EC2 Cloud was also used as a reference platform. Their client devices were all located in the UK and connected to the Cloudlets using VNC. Clinch et al. [11] find that when experimenting on the reaction time of users playing a whack-a-mole game on the thin-clients, the network distance did play a determining factor in the accuracy of the user's interaction with the game. However, they conclude that there is no definitive answer to how services should be localised as there are multiple factors affecting the decision such as network latency, host hardware/software configuration and interaction intensity of the application.

Shea et al. (2013) [61] argue that delay tolerance depends on the game genre. Action games that require fast reactions can provide a good experience when latency is below 100ms. For role-playing games, the tolerable latency can be up to 500ms and for real-time strategy games, it can be up to 1000ms. Based on the above, we can also argue that turn-based games can tolerate in excess of 1000ms latency since interaction between the players is not happening in real-time. However, the important thing to consider here is that a good QoE requires low latency. Furthermore, as presented in the same article, image quality is dependent on bandwidth. More bandwidth provides better Peak Signal to Noise Ratio and image quality similar to local rendering, while less bandwidth creates more blurred images with less detail and definition.

Because latency and bandwidth directly affect the QoE in terms of interaction delay and image quality respectively, we can also deduce that using image compression techniques to improve image quality over low bandwidth connections will have an adverse effect on the latency. Similarly, simplifying the rendering and image transmission may degrade image quality but also reduce latency. Therefore there is a need for optimised solutions to this problem. OnLive is a Cloud gaming service provider and in their case, the encoder (H.264/MPEG-4) latency is reported to be 4ms (OnLive,

2013) [51]. However, the trade-off as reported by Leadbetter (2010) [38], is that games cannot run at their best image quality since the encoder takes away a lot of the fine detail in images. This low latency is also made possible by the relative low resolution of 1280x720 in which games are rendered, however, OnLive claims that once higher sustainable bandwidth becomes widely available, the games will render at 1920x1080 (OnLive, 2014) [52].

It is also worth noting that OnLive can offer its services to mobile devices making it possible for demanding games to run on hardware with limited capabilities. However bandwidth consumption is a problem for mobile devices, especially when using 3G and LTE connections where data limits are a common practice for preserving network performance and keeping the network costs low. Traffic management is an ongoing problem on the Internet and as we recently saw with the Netflix and Comcast peering agreement, prioritisation of traffic can be costly and network providers may choose to ask for extra money for enhancing the performance of a service that uses their network. The focus is therefore placed on modern network technologies and service deployment models that will enhance the performance of networks through adaptive QoS provisioning and dynamic traffic management.

2.4 Critical Summary

This chapter presents the evolution of task offloading from mobile devices. While early solutions for mobile task offloading did not enable the development of thin-client mobile computing, they set the foundations by providing insights and demonstrating how certain tasks can be assigned to a local surrogate node in order to speed up the execution of software on the mobile device or enable the execution of software that would otherwise be impossible to run on the limited on-package resources of these devices. One of the main drawbacks of the early offloading technologies was that local surrogate nodes had to be deployed at the LAN and hence the responsibility for maintenance and support fell on the administrators of the LAN. In the context of mobile computing, where such wireless networks could be a free service from shops and small establishments, the above approach presented challenges and difficulties when deployed in this new setting. Furthermore, for larger networks, the surrogate would have to be powerful server-class equipment which further added to the complexity of deployment and maintenance.

Cloud technology addresses these drawbacks by providing a large amount of resources that are accessible to everyone albeit not localised and therefore the performance of the connection between the client and the server plays a significant role to the performance of a service. Modern network technology makes it possible to run complex applications on the Cloud and access them remotely via thin-clients. However, such service provisioning is limited to cases where network connectivity is reliable and adheres to the minimum requirements of the service. Regionalised Cloud datacentres help alleviate such problems and deliver high-performance services to their clients, however, when applications require a very high level of interaction and bandwidth, regionalised datacentres do not present a solution that is fine-grained enough to support interactive services in a mobile environment. In such cases, Cloud technology may benefit from the approach of the earlier attempts for mobile task offloading by employing a solution that is more fine-grained than existing regional datacentres but also easier to deploy and manage than localised surrogates within LANs.

In the next chapter, we analyse the structure of the Internet, discuss its evolution and present some of the newest technologies for managing traffic and improving network performance by distributing content to multiple locations and delivering it to clients from the datacentres located within or peering with Internet Service Provider networks.

Chapter 3 The Evolving Internet: Structure, Framework and Implications

We often define the Internet as a network of networks which has a mesh structure; however, this often leads to the misconception that every network is perfectly interconnected with all the other networks around it. In reality, the Internet is a partial mesh which has a layered topology where many networks are not directly connected to each other but instead use third party networks to achieve connectivity. In essence, point to point connectivity is not always guaranteed for end-to-end communications. In the following subsections, we analyse how the Internet is structured. We will also look at modern technologies for content distribution and delivery. Finally, we will examine the impact of virtualisation technology on the Internet.

3.1 Background on Internet Transit and Peering

As mentioned previously, the Internet consists of multiple networks that are partially interconnected. These networks are divided into three tiers, depending on how they interconnect with other networks and what coverage area they have (Berg, 2008) [5]. Tier 1 networks are those that have very wide coverage areas. They have their own backhaul connections with multiple Points of Presence (PoP) around the globe. Tier 1 networks peer with each other via their PoP in order to access remote networks. Tier 2 networks may also peer with each other but they also purchase transit connections from Tier 1 providers. This means that Tier 2 networks have to rely on a Tier 1 and their peers, to access networks that are not directly accessible to them. Tier 3 networks have to rely on transit connections with Tier 2 and Tier 1 providers to get all of their network access. The diagram in Fig. 3.1 illustrates the tier structure of the Internet.

Peering can be either public or private. For public peering, multiple networks converge at Internet Exchange Points (IXP) where they interconnect with each other. In a single IXP, one entity can connect to multiple other networks, thus minimising peering costs. On the other hand, a private peering agreement between two or more entities may be preferred when dealing with high traffic volumes or when security is important. Private peering costs are higher but the performance is guaranteed and it is a more controlled environment.

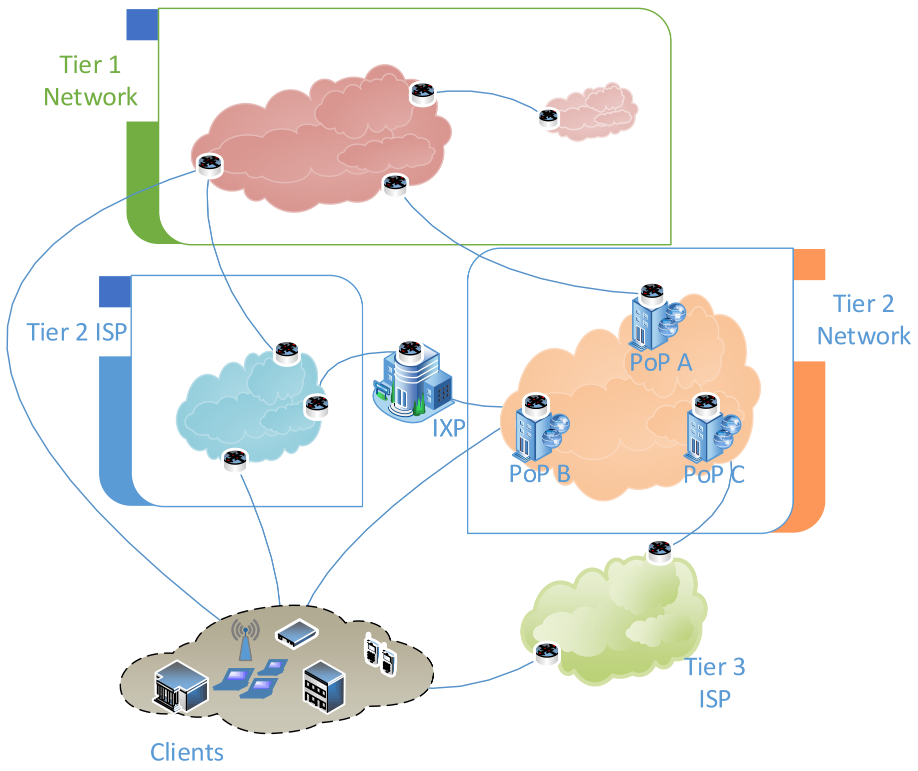


Figure 3.1 Tiered Internet structure and peering/transit connections.

To summarise, peering is the practice of interconnecting networks at public or private locations with the aim of giving end-to-end access to each other for them and their clients. As a result, costs can be reduced and performance can be increased by bypassing higher tier providers that would act as a backbone. However, it is infeasible to use peering to connect every network with another; therefore, Tier 2 networks will always have transit connections to Tier 1. Table 3.1 includes some of the peers that can be found in the London Internet Exchange (LINX) and Moscow Internet Exchange (MSK-IX) which are two of the largest in the world, to demonstrate how global connectivity is achieved.

Table 3.1 Sample from LINX and MSK-IX peering records registered in www.peeringdb.com

LINX			MSK-IX		
Peer	**Traffic**	**Ratio**	**Peer**	**Traffic**	**Ratio**
Akamai	1 Tbps	Heavy Outbound	Microsoft	1+ Tbps	Balanced
Facebook	1+ Tbps	Mostly Outbound	Google	N/D	Mostly Outbound
Janet	100+ Gbps	Mostly Inbound	Megafon	0.5 - 1 Tbps	Balanced
OpenDNS	10-20 Gbps	Balanced	Rostelecom	1+ Tbps	Mostly Inbound
Symantec Cloud	5-10 Gbps	Balanced	Yandex	0.1 - 0.2 Tbps	Mostly Outbound
BT	100+ Gbps	Mostly Inbound	Verisign	N/D	Balanced

Transit connections represent a customer-provider relationship in the sense that it is the lower tiers that use them to get access to networks of the upper tiers and everything they have access to. A higher tier provider will charge the lower tiers for transit connections. The cost is calculated on a price/Mbps. Although the transit costs have steadily been declining for a decade, peering costs are typically lower. It is also the fact that most IP traffic is asymmetric, meaning that small requests generate large responses, which makes transit connections expensive. For example, if we imagine a Tier 3 ISP with a few thousand clients within a city, we can see that their traffic ratio will most likely be heavily inbound if their clients are homes and small businesses with very little content published. However, in the opposite scenario, if the clients are content providers or aggregators, then it makes more sense to create a Tier 2 network with peers to other Tier 2 networks and transit to Tier 1 for global backbone connectivity.

To better understand peering and transit as well as the tiers, we can look at the example of British Telecom (BT) from LINX in Table 3.1 above. BT is considered a Tier 1 provider with multiple PoP around the world. They are the primary provider for the UK and have free access to the Internet via their own infrastructure and reciprocal peering agreements. We can see from Table 2.1 that the traffic ratio for BT is mostly inbound. This means that for BT, most traffic flows terminate in their network. We know that BT also is an ISP that provides connectivity to homes and businesses in the UK, as well as the peering and transit agreements they have with other networks. So this tells us that clients in BT's network are mostly consuming content or in other words, BT's network is not used so much to make content available to other networks. On the other hand, a network such as Akamai, also present in LINX, has heavy outbound traffic. Akamai, is a Content Distribution Network and as will be explained in a following section, their main traffic flows are outbound due to the IP traffic asymmetry (small requests generate large responses) and because they do not offer ISP services to consumers. Finally, we see that OpenDNS has balanced traffic flow ratio because the DNS service does not follow the asymmetry rule. One request gives one response which results in a balance between inbound and outbound flows. Having looked at the above examples we can see that networks on the Internet can be classified as content providers/distributors, transport networks and consumers. Any single network can be a combination of those three and depending on which role carries more traffic volume, the tier level of the network can be adjusted by changing their peer and transit connections.

In summary, the Internet's fabric is only a partial mesh with very distinct hierarchical tiers. This distinction comes from differences in performance, coverage and peer connectivity in each tier, with the bottom tiers having the smallest coverage and relying on transit connections to the upper tiers in order to achieve global connectivity. Peering can be either public or private, with public IXPs providing managed locations for networks to deploy their PoP while private locations are used for dedicated, point-to-point connections between two networks for improved performance and security when necessary.

3.2 New Internet, Cycle Peering and Mesh

In recent years, the structure of the Internet is changing. As explained by Woodcock, (2003) [71], from a partial mesh, the topology is changing to a ring formed by Tier 2 networks on the periphery and Tier 1 networks in the centre. Due to this shift in the topology, the Tier 1 networks in the centre are less utilised due to the abundant peering connections among Tier 2. Driven by the economics of interconnection, Tier 2 providers are forming peering agreements with each other in an effort to improve the performance of their networks and decrease their transit costs from Tier 1 networks (Cook, 2002) [12]. The result of this trend is that Tier 1 networks, now residing in the centre of this ring topology are becoming dark areas with less and less traffic going through them, hence the term cycle peering. This result can be seen from as early as 2008 when the Internet traffic began to slowly bypass the U.S. which until that time was host to some of the biggest Tier 1 networks (Markoff, 2008) [45]. As we see today, the U.S. still holds the highest percentage of IPv4 allocation with IPv6 adoption also the highest for that region; however, emerging markets are showing a change of balance for the future (CAIDA, 2014) [6].

We are now slowly moving towards a mesh structure on the fabric of the Internet with Autonomous Systems peering with each other in order to minimise costs and improve performance (Filippi, 2014) [19]. However, the complexity of mesh networks in terms of routing and management is slowing down the adoption of this topology. Another reason for the slow adoption of mesh networks is that despite their resilience against faults due to their degree of self-healing, it becomes harder to maintain control of traffic and block unwanted access. Economics also play a role because transit connections from higher tier networks are revenue generators and some operators have attempted in the past to restrict traffic through certain routes that will increase their revenue (Valancius, 2011) [66]. As mentioned previously, the Internet already has a partial mesh structure with Tier 2 and Tier 1 networks peering with each other while using transit connections to gain access to networks that are not directly attached. The difference in the new mesh structure is that as the ring of peripheral networks increases, the peering interconnects between those networks are becoming more extensive, thus avoiding transit connections to Tier 1.

In summary, as Tier 2 networks are growing in size and numbers and the oligopoly of Tier 1 networks is becoming redundant, the Internet will slowly evolve into a full mesh topology with Autonomous Systems peering with each other in various PoP around the world. The advantages of this evolution include lower costs for the networks, increased performance and higher reliability. Internet caching technology is using this connectivity to its advantage by placing content caching datacentres within networks in order to maximise performance for them and their peers, while at the same time, building caching Autonomous Systems that peer with multiple networks in order to better distribute content.

3.3 CDN and Edge-caching

Web caching technology was created in order to maximise network performance by reducing the amount of traffic that crosses an Autonomous System's boundaries (Huston, 2009) [29]. This also drove down costs for the Autonomous Systems, especially for content that had to be fetched over transit connections. The performance advantage comes from the fact that a regional or local cache can reduce the load on a content publisher's servers by keeping copies of the content and serving client requests. It is a form of load balancing by distributing the content and regionalising access requests. The second factor that improves performance is that regional caches are often closer to the clients than the content publisher's servers as illustrated in Fig. 3.2. In effect, by shortening the path between server and client, the load on routers is reduced, the transit costs are reduced and as a result, the performance of the network increases.

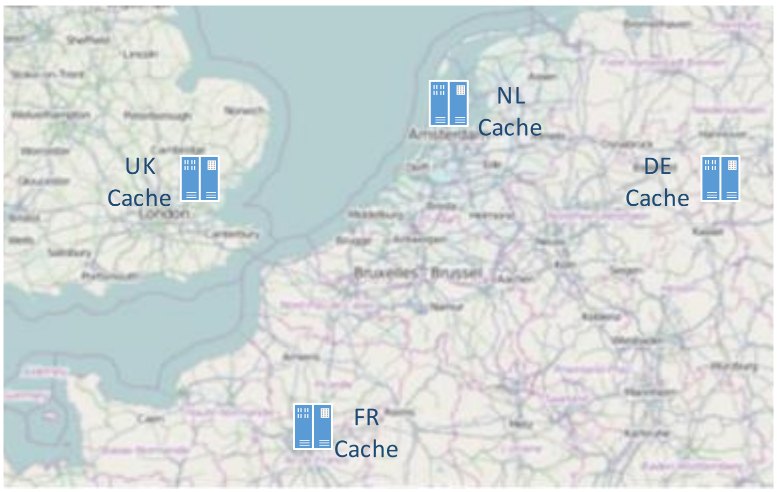


Figure 3.2 Regional localisation of web-caches.

At present, caching has evolved into what is now called Content Delivery Networks. Like web caches, CDNs host web content such as text, images and videos. The main difference with web caches is that a CDN is a distributed system operated by an organisation with regional PoP. The CDN operator is paid by content publishers to deliver their content to end-users. Because the CDN is a distributed system, load balancing is more fine-grained. It can also completely negate the need for servers on the publisher's side since the content can be made available directly from the CDN. This form of distributed caching localised at the edge of an Autonomous Systems is called Edge-Caching and it is illustrated in Fig. 3.3. The rapid growth of multimedia content consumption has made caching at the last mile more important and many companies are now involved in CDNs and Edge-Caching. Finally, the increasing popularity of heterogeneous devices on the Internet such as laptops, tables and Smart Phones, has created the need for Mobile CDNs where, among other factors such as client location and network performance, the capabilities of the client's device are taken into account in an effort to deliver customised content suitable for viewing in each device. The development of CDNs along with the incentive of creating peering connections between networks has created a new topology map for the Internet, as described by Labovitz et al. (2011) [37] and presented in Fig. 3.4.

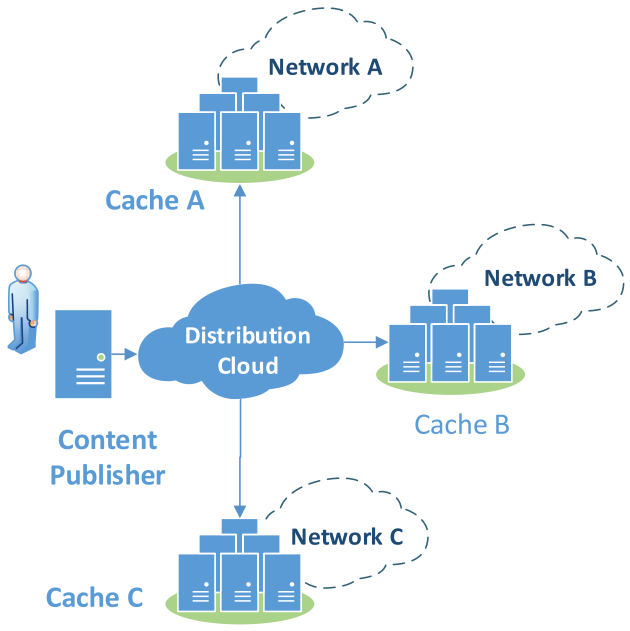


Figure 3.3 **CDN topology for Edge-Caching.**

In summary, web caches have existed for a long time with the purpose of balancing the load, decreasing the traffic volume that crosses network boundaries and improving the performance of services. CDNs are now used for more fine-grained caching where content is hosted at the edge of a network with the purpose of servicinng its own clients better as well as clients of its peers. The distributed nature of CDNs makes them more efficient at delivering content and sometimes negates the need for a publisher to own servers. Another advantage is that the technology is capable of delivering content by taking into account factors such as the client's device characteristics and customising the content to suit the device.

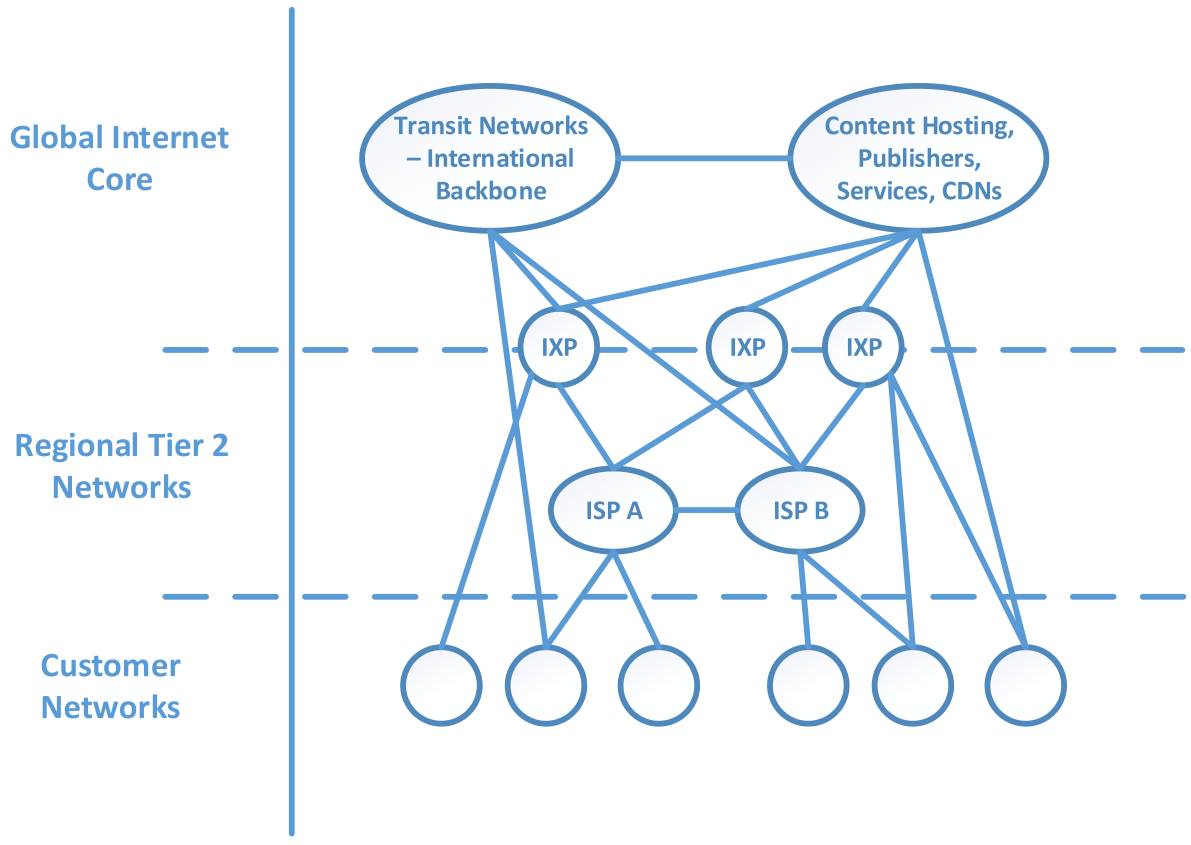

Global Internet Core
Transit Networks – International Backbone
Content Hosting, Publishers, Services, CDNs
IXP
IXP
IXP
Regional Tier 2 Networks
ISP A
ISP B
Customer Networks

Figure 3.4 **Evolution of Internet structure using CDNs and Edge-Caching.**

3.4 Performance

In analysing the performance aspect of networks it is worth noting the work by Krishnan et al. (2009) [36]. They present a set of data gathered by Google's CDN and explain how Google's latency-based redirection method is not always effective due to path inflation resulting in much higher experienced latencies than the reported nominal for a particular Autonomous System (AS). To achieve the lowest latency possible for a client within a network, Google usually redirects the requests to the closest CDN geographically. However, Krishnan et al. discovered that clients in geographical proximity often experience widely different latencies. They explored the underlying causes for path inflation and present them as follows:

Lack of peering: When all available paths from an AS to a CDN are via third party networks despite the AS being geographically close to the CDN.

Limited bandwidth capacity: This can occur either due to circuitous path or lack of bandwidth availability at the peering facility between the AS and the CDN.

Routing misconfiguration: This occurs when network devices are misconfigured and although traffic in one direction goes through the lowest latency path, the response traffic in the opposite direction takes a circuitous path, thus inflating the response time.

Traffic engineering: This occurs when the AS and the CDN do not directly peer and for traffic engineering reasons the connections from the AS to the CDN are configured to pass through third-party networks that do not offer the shortest communication path.

These causes can be addressed directly by reconfiguring the networks or by setting up new peering agreements and new CDN PoP, however they are not something that can be easily fixed by employing SDN solutions because SDN implementations do not necessarily extend outside the boundaries of an AS and therefore they can only address a limited number of these problems.

From a Cloud computing perspective, such network problems can become more important, especially when considering high-performance, interactive services. Studies show that networks pose a bottleneck to high-performance Cloud services and therefore network QoS capabilities are of paramount importance for Cloud computing (Duan, 2012) [16]. This is not only limited to the service networks that connect a user to the Cloud but also the networks that compose the fabric of the Cloud. Ultimately, a Cloud service is a composition of both cloud and network services and its performance is directly affected by them. Consequently, Cloud services that span multiple datacentres in different geographical locations may experience the problems described above. To resolve this problem a new perspective to service delivery is necessary.

In the Service-Oriented Architecture (SOA) model, a composition of smaller, simpler and sometimes heterogeneous services are invoked through their independent interfaces to provide a larger more complex service (Erl, 2005) [17]. SOA enables the virtualisation of resources in the form of services and the subsequent interaction of these services in order to provide a more complex solution to a customer's requirements. A paradigm of SOA application is Cloud computing, where various modules and services running on different parts of the infrastructure, form a complete business solution for a client. The same service-oriented principle applied to networks, supports the virtualisation of network resources running on an underlying infrastructure and essentially decouples the infrastructure from the network services.

This type of network virtualisation applied Internet-wide enables multiple network providers to compose heterogeneous virtual networks independent of each other while all sharing the same physical network infrastructure. Therefore, Service-Oriented Network virtualisation divides the role of an ISP into Infrastructure Provider and Network Service Provider. On one hand, the Infrastructure Provider is responsible for building and maintaining the physical plane of the network, while on the other hand, the Service Provider creates virtual networks that provide end-to-end connectivity. Because network virtualisation allows a single ISP to control end-to end connectivity, it also gives them the ability to define QoS provisioning across the path of the communication and to choose which physical networks will be used to carry the data. Fig. 3.5 illustrates the SOA environment.

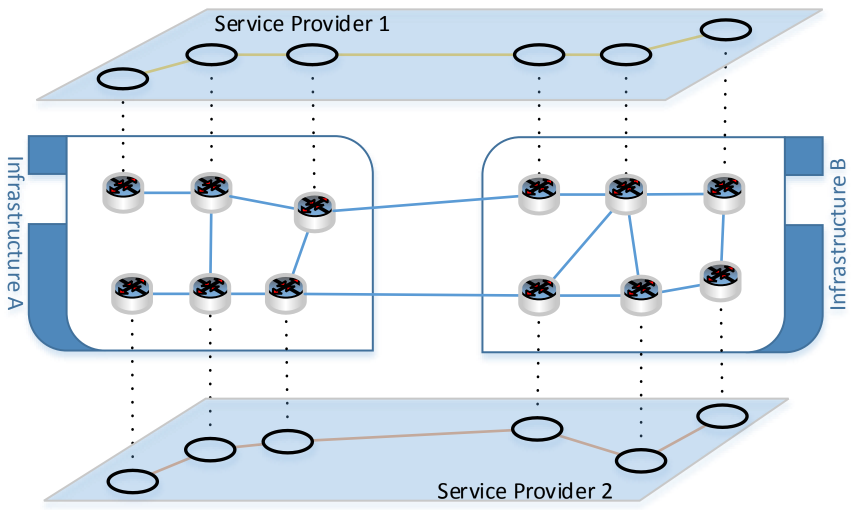


Figure 3.5 Service-Oriented architecture based on fixed infrastructure networks.

When it comes to composing services over a virtual network, Duan et al. [16] outline are various schemes that can be employed. A Static Design Composition predefines which component services will be used by a composite service. Manual Composition relies on the user to select and compose the services as opposed to Automatic Composition which requires no user input. Finally, dynamic composition uses a set of runtime parameters taken by component services to determine which instances are best used for delivering a composite service to a particular client. Such inputs can be the computing cost, network cost and service availability. To evaluate a composite service as proposed, we can use execution time, computing and network cost and composition sustainability in case of component failure.

Some of the challenges in composing services over wide area networks include QoS in the network infrastructure, device heterogeneity in the infrastructure, semantic data modelling such as service discover/selection and semantic metadata such as service taxonomy. Purely from a networking perspective, the two questions that arise from the above are: 1) how to create the best possible service path while meeting services QoS requirements, and 2) how to balance the loads from the services to achieve the best possible utilisation of resources. There are various frameworks that attempt to address these two problems. Lee et al. (2013) mainly divide them into those that focus on service response times and those that focus on load balancing. SATO, proposed by Cheng et al. (2006) [9] and ContextWare, proposed by Ocampo et al. (2005) [50] are two examples of response-centric framework, while QUEST, proposed by Gu et al.

(2003) [23], is an example of load balancing framework. The state of art in the area of service path selection is QALB (Lee at al., 2013) [39] which is a framework that balances response times and network load to achieve the best performance possible. Results published by Lee et al. show that QALB can achieve better Request Success rates compared to other path selection schemes. It also achieves lower QoS Violation rates; however we see that on both cases, as the number of client requests increases, the QoS Violation rate can reach almost 40%. Therefore, QoS is never guaranteed and the level of QoS provided is specific to the scheme that a particular service provider has selected. What is a determining factor for the above is the infrastructure to which the provider has access and how that infrastructure performs. Therefore, some schemes may perform better than others but ultimately, the effective QoS for a service depends on the current load on the infrastructure.

To facilitate some of the network performance requirements of distributed datacentres and Cloud computing, several technologies have emerged in recent years for the purpose of improving networks services, managing traffic and using network resources more efficiently.

3.5 Software Defined Networking and Network Function Virtualisation

In this section we will look at how networks are evolving towards Software Defined Networking (SDN) and Network Function Virtualisation (NFV). Before looking at NFV, it is best to explain the concepts of Capital and Operation Expenditures (CAPEX and OPEX respectively) which are the two key-words when discussing NFV, and see how they are providing a driving force for the deployment of these technologies. A network operator's OPEX includes operational costs to run and maintain their network. Included in these costs are the specialists needed for configuring and maintaining the network as well as replacement hardware to cover failures or upgrades. The CAPEX includes costs for expanding a network such as buying new equipment to cover a new region or a major upgrade of the equipment serving a particular area. The advent of Cloud technology has provided the ground for virtualising network functions by implementing proprietary network equipment in software and consolidating it within a Cloud environment. Industry standard, high volume servers can be used for Cloud deployment thus negating the need for specialised equipment. This drives OPEX costs lower as specialised hardware maintenance is no longer necessary. This advantage gives the opportunity to operators to deploy new services faster, at a lower cost and with the flexibility to scale up or down more easily according to client demand. It also gives smaller operators a better chance at competing due to the lower risk involved in the deployment. The above also translates to lower CAPEX for operators wishing to expand their networks.

In summary, the aims of NFV according to the NFV architectural framework standard (GS NFV 002) are to increase capital efficiencies compared to dedicated equipment implementations via the use of Commercial-off-the-shelf hardware, to improve the scalability and decouple functionality from location by means of deploying virtualised functions remotely and reusing a pool of resources, to increase the innovation through software-based service deployment, to improve operational efficiencies by making use of the elasticity of resources inherent in Clouds and to standardise interfaces between network functions and infrastructure so that a network provider can choose and mix decoupled elements from different vendors. Although NFV and SDN are two technologies complimentary to each other, they do not present a mutual requirement for deploying one or the other. In other words, it is not necessary to deploy NFV in order to deploy SND and vice versa.

Software Defined Networking has three different deployment models: Network Virtualisation, Evolutionary and OpenFlow. The main concept of SDN is the decoupling of the plane from the physical hardware. In simpler terms, SDN creates a management layer above the network hardware which is used to adjust traffic routes and the performance of the network. Virtual LAN (vLAN) is one technology which can be used as an example of a control plane that defines virtual networks in a switch and allows the partitioning or grouping of switch ports so that different LANs can use a single switch without experiencing problems in terms of broadcasting and multicasting packets. In essence, many LANs can use a single switch and still be isolated from each other. The main deployment models of SDN are defined as follows (Nolle, 2013) [49]:

Network Virtualisation Model: The network virtualisation model is aimed at eliminating the problems that exist between physical network partitioning and vLANs. It achieves this by implementing interfaces at the Hypervisor that create vLANs which operate in tunnels over traditional Ethernet. The advantage of this approach is that the physical network is unaware of any virtual partitioning while virtual networks within a Cloud can maintain their isolation. This means that multi-tenant Clouds are supported without making changes to the networking hardware. One disadvantage is that since the vLANs operate within tunnels, they appear to the switch as ordinary traffic and leave no opportunity for traffic prioritisation. Because these virtual networks are created by the software that runs on the Cloud stack, this implementation can only link virtual machines with each other and anything outside these tunnels including physical devices cannot be connected.

The Evolutionary Model: The Evolutionary Model takes the virtualisation model one step further by allowing the differentiation between vLANs at Layer 3, thus enabling the routing of vLAN traffic tunnels across different IP networks. The aim is to extend the connectivity of the virtualisation model so that routers can be used to interconnect vLANs residing in multiple physical switches at different locations. A simpler way to look at the Evolutionary Model is to consider it as a Layer 3 extension to Layer 2 vLANs the same way that IP addresses are a logical extension to the physical addresses in the OSI. This extension of the virtualisation model is made necessary by the fact that vLAN address space is limited to 4094 addresses which is not enough in large Cloud implementations with thousands of tenants and datacentres in different locations interconnected via IP networks. The Evolutionary Model and more specifically the Virtual Extensible LAN VXLAN model allows for the encapsulation of vLAN traffic before it leaves the switching domain of a datacentre and enters the IP network. The packets leaving a VM with a destination and source IP address as well as a VXLAN ID are further encapsulated with the IP address of the destination datacentre. On the receiving end, the gateway decapsulates the packet from the top header IP address and looks at the VXLAN ID to find in which vLAN the packets should be forwarded to. The packets are decapsulated again after entering the vLAN at which point normal IP and MAC addressing is used to reach the destination VM.

OpenFlow Model: Perhaps the most associated model with SDN is the OpenFlow model. OpenFlow employs a central controller that programs each network device's forwarding table as opposed to using discovery mechanisms. This gives a fine-grained control over how a network is segmented and how traffic is managed. The disadvantage is that network devices have to support the OpenFlow model natively in order to be compatible with the instructions issued by the central controller. The advantage is that the central controller has a complete view of the network fabric and is in position to decide on how traffic should be forwarded and elastically distribute resources to traffic that requires a high QoS. The achievable results are close to 100% utilisation of network resources as opposed to near 40% utilisation of the standard model with minimal risk of network overload as the central controller is able to dynamically reroute traffic and reallocate resources.

The programmability models employed by SDN can be described as reactive, proactive and predictive. The reactive model mostly associated with OpenFlow keeps track of traffic flows on the network and constantly adjusts routes and QoS based on the observed conditions. The proactive model improves on the reactive one by monitoring conditions on the network and making predictions on where and when potential problems may arise. Changes are made to traffic routing before performance problems arise. The predictive model keeps a record of historical data and identifies trends and patters in the traffic flows. This makes it possible to provision resources and reconfigure the network in anticipation of periodic changes in utilisation. The need for proactive and predictive models stems from the fact that reactive models such as OpenFlow are incapable of keeping up with rapid traffic changes that often occur in highly active environments such as datacentres.

To conclude, SDN and NFV technology can assist in traffic management and cost reduction for network operators and service providers. Both technologies are still currently in development and in some cases they (such as Google's CDN), they are already deployed. Perhaps the most important observation to be made is that we are moving towards a virtualised world, where even networks are managed using Cloud technology and network operators are considering Cloud datacentres within their infrastructure. Looking back at Cloudlet deployment models, we can observe that as network operators build Cloud datacentres SDN and NFV functions, they may also look at the possibility of offering some of their Cloud resources to their clients for MCC purposes. However, before considering localised datacentres as the solution to MCC, we need to understand the concept of traffic localisation and how traffic could be managed in such way that constant use of MCC will not congest networks on a global scale.

3.6 Economic Traffic Management

One of the key aspects to investigate within the context of Cloud services is the impact of the extra traffic generated on the Internet from the client connections, especially when displaying media-rich content. As explained, network providers are pursuing peering connections as opposed to transit connections in the backbone networks. They are also interested in localising traffic as efficiently as possible in order to reduce inter-AS traffic and decongest peering interfaces. To calculate the costs of inter-AS traffic, network operators consider the 95th percentile of traffic samples in order to eliminate traffic spikes during peak times from becoming a major determining factor of the pricing. The inbound/outbound traffic between two operators is compared at the end of each month and the operator with the higher inbound traffic is charged.

Economic Traffic Management (ETM) assumes that operators will voluntarily participate in a scheme that reduces costs incurred by the above billing method, while at the same time decongesting their networks by employing locality promotion. The idea behind locality promotion is to contain traffic within a domain. Data that has to be fetched over multiple networks causes increased costs for all the involved parties, therefore localising data whenever possible will reduce the amount of traffic exchanged between domains. The benefit of locality promotion is not restricted to cost reduction. Connections over long network distances (high hop counts), consume more routing and switching resources for transmission and this adds to the congestion on the backbone interfaces. Therefore locality promotion can contribute to enhancing the QoS of a network and the user's QoE. The main terminologies concerning ETM are presented in Table 3.2 along with their definitions.

Table 3.2 ETM terminology synthesis table.

Economic Traffic Management

Inbound Traffic	Traffic that enters network boundaries in the form of remote client requests to local servers or remote server responses to local clients
Outbound Traffic	Traffic that leaves network boundaries in the form of client requests to remote servers or responses from local servers to remote clients
Traffic Localisation	Keeping traffic within a network by caching frequently-requested content locally
Swarm	The number of users exchanging data with a particular server

There are two ways to deploy ETM. One way is to create an overlay that identifies where a particular piece of data exists and rank it based on the network distance from the user requesting it. In a P2P example, if we consider a user downloading a particular file, parts of this file may reside in nodes on the user's current domain while other parts may be residing in third party networks. The parts that are local to the user's domain or to a domain that is a peer receive a higher rank and are preferred as sources compared to parts that reside in domains reachable over a transit connection. Fetching data from the local network and its peers results in better QoS for the user's service and also reduces the transit costs for the provider. The other way is to employ ISP-owned peers, which is effectively a form of caching this data within a domain. ISP-owned peers appear to the user as ordinary peers with the difference that it has dedicated resources for this task and therefore are more effective at seeding data. For a more general example we can consider CDNs that peer directly to domains and distribute content by caching it either locally at the providers or via peering at IXPs.

The above information is presented in the article *"An Economic Traffic Management Approach to Enable the TripleWin for Users, ISPs, and Overlay Providers"*, where Hobfeld et al. (2009) [26] also present the results of their study in applying ETM for BitTorrent. The first identified parameter for making ETM effective is the number of clients within a domain that engage in data exchanges that are subject to locality promotion. In their experiment they consider the size of peer swarms using P2P within a domain. They determine that in order for ETM to be effective in achieving a substantial cost reduction, it has to tackle swarms of all sizes. In a simulation study, they divided a swarm of 50 users into networks A and B of 35 and 15 peers respectively. In network B, locality promotion has resulted in 15% reduction of inter-domain traffic and by adding an ISP-owned peer to network B; the ingress traffic was further reduced by 45%. The addition of an ISP-owned peer in network B also caused an increase in ingress traffic by 55% for network A.

But ETM does not always bring benefits for network operators and users. For example, Piatek et al. (2009) [53] argue that the QoS may actually degrade for a user when considering traffic localisation applied to P2P traffic within an AS that allocates asymmetric bandwidth for its users. Additionally, P2P traffic localisation may sound as a good idea from the perspective of residential ISPs (where end-users connect) but from the perspective of transit networks, it can be quite damaging to the profitability. Another problem is that P2P clients typically have only a few concurrent peers at any one time and a very small number of them are found within the same ISP.

Perhaps the most extensive analysis of the impact of P2P traffic localisation on ISP profitability is presented by Seibert et al. (2012) [60]. They examine different pricing and charging models along with different localisation models and summarise their findings in a series of insights. Some of the most significant insights are analysed below:

*"**Insight #2:** Some residential Autonomous Systems will actually lose profit when they localize traffic. This is due to these Autonomous Systems also being transit providers for other residential Autonomous Systems. For these Autonomous Systems, P2P traffic that was previously downloaded from clients in customer Autonomous Systems decreases due to localization and in turn revenue decreases. Therefore, they have little incentive to localize traffic."*

Insight #2 is telling us that localising traffic may actually reduce profitability for transit networks such as Tier-1 and Tier-2. In general, any network that heavily relies on profits made from its transit function for other networks, will suffer losses if other networks start using traffic localisation. Since most ISPs are also transit networks, it makes little sense for them to localise traffic. However, it may be of benefit to smaller ISP networks.

*"**Insight #3:** Content availability plays a crucial role in determining the effectiveness of localization. Due to churn, peers will often need to re-download content from outside the AS. However, when assuming persistent content, most Autonomous Systems can reduce losses twice as much."*

To make localisation successful, content needs to be available within one network and its peers. This can be achieved more effectively when the Autonomous Systems have similar clients in terms of language and culture. This way, we can ensure that content which is most likely to be requested by clients, is already residing within the network. It becomes easier to have such content localised when it is persistent (non-dynamic) and therefore such content can help maximise the benefits of traffic localisation.

*"**Insight #8:** Pricing scheme has a large impact on the effectiveness of savings. As the maximum pricing model ignores one direction of traffic, reduction in the other direction*

does not result in a reduction of cost. The average pricing model does consider both inbound and outbound traffic and thus an AS could benefit both if it or some other AS localizes traffic."

*"**Insight #9:** Contrary to the average pricing model, for the maximum pricing model it is not sufficient that few Autonomous Systems localize traffic to reduce cost. Even if the largest Autonomous Systems start deploying localization schemes, overall loss reduction will be very limited."*

This is perhaps one of the most important insights. The pricing model is the key factor that determines whether or not traffic localisation can reduce costs. There are two pricing models used to determine monthly charges between network operators. The "maximum" charging model compares the inbound and outbound traffic for the whole month and chooses the higher of the two as the determining factor of the monthly bill. Therefore, reduction in one-way traffic may not always reflect to cost reduction. However, when the pricing model considers the average of inbound/outbound traffic, then localisation can have a more immediate effect to the cost.

*"**Insight #10:** Many Autonomous Systems will achieve more profits through preferentially directing traffic to customers and peers rather than localizing traffic. Therefore, P2P traffic localization is not always the best choice for all Autonomous Systems."*

Completely localising traffic in a "naïve" and "obsessive" manner does not necessarily bring any benefits. Profits may actually increase by smarter methods of directing traffic such as preferring clients from peering networks. Since peering connections are free, there is no direct cost involved in treating peers as "local" when directing traffic. Consequentially, each peering ISP will have a much bigger swarm to act as the local cache.

*"**Insight #11:** While business-relationship based policies may locally be the best strategy for some Autonomous Systems, they can have a negative external impact on other Autonomous Systems. Furthermore, as the best local strategy of an individual AS is chosen in isolation of others it does not turn to be the best possible choice when all Autonomous Systems deploy their own best local strategy."*

Ideally ISPs will have to agree on a common practice that will benefit all of them. This way traffic localisation may bring benefits to everyone either in the form of cost reduction or network decongestion.

Perhaps the most important aspect of ETM is not economic from the financial perspective but rather from the network traffic congestion perspective. As argued by Piatek et al. (2009) [53] and Seibert et al., (2012) [60], ETM is not particularly effective at increasing profits or reducing costs in the real world. However, by taking into account the rapid growth of traffic demand, we see that the economic balance of the Internet is changing and ETM may be able to offer a long-term solution to sustainable growth without increased CAPEX and OPEX. Since cyclic peering is becoming a reality and the peering networks are not bound by billing agreements, ETM can provide a solution for more efficient management of peering bandwidth rather than cutting costs or increasing profits. More efficient traffic management across Autonomous Systems along with technologies such as SDN and NFV can drive down the cost of investment for new equipment to meet increased traffic demands. Therefore, when it comes to MCC services, we could consider ETM mechanisms as a means for dynamically deciding when and where a user's service may be localised. This way, we ensure that traffic management aspects are taken into account instead of obsessively localising services according to user location without considering their impact on the networks.

3.7 5th Generation Networks Y-Comm Framework

The increasing popularity of mobile devices that feature multiple network interfaces has driven the progress in the subject of seamless vertical network handovers. A vertical network handover is the process in which a device switches between two different network technologies such as Wi-Fi and LTE while maintaining connectivity and the process is transparent to the user and the applications. A horizontal handover, on the other hand, is a handover process between two access points of the same technology. In either case, the handover event may occur either within the same administrative domain or across two different autonomous domains. Fig. 3.6 illustrates a vertical handover scenario.

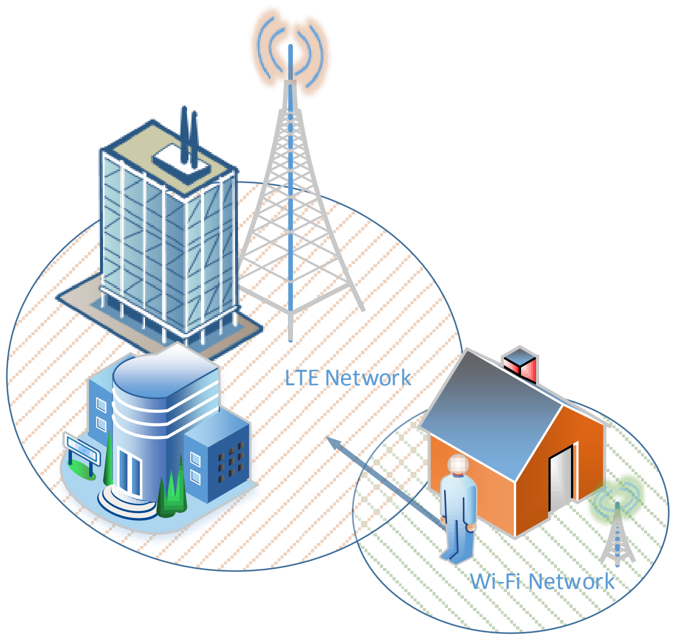


Figure 3.6 Vertical handover scenario between Wi-Fi and LTE networks.

To provide seamless vertical and horizontal handovers, Mapp et al. (2006) [43] proposed Y-Comm. Y-Comm splits the Internet in two separate entities named Core and Periphery. The Core network is the part of the Internet where fast connections such as optical fibre exist and interconnect individual Autonomous Systems. The Peripheral network consists of slower networks such as Wi-Fi, LTE and ADSL. The two networks join together at Core End-Points where the Peripheral networks gain access to the Core networks and their services.

Mapp et al. argue that the OSI model is no longer sufficient for dealing with connectivity in the modern era of mobile users and devices. The foundation of the argument lies in that the OSI was created as a theoretical framework for communication between two end-points in a static network and is inherently incapable of dealing with mobility and its consequences to the QoS. Although we could argue against this by presenting various solutions that have been implemented at the network and transport layers of the framework, it should also be noted that these solutions are often add-ons and refinements to the OSI layers which is often not the most efficient way of introducing such functionality. The architectural framework proposed by Mapp et al. (2007) [42] takes into account user mobility, QoS and security for providing network connectivity as well as services to users. The framework consists of two parts: one for the Core network and one for the Peripheral networks, both of which join at the two bottom layers forming a Y-shape hence the name.

Constant mobile connectivity plays an important role within the scope of this thesis and vertical handover events are in effect trigger events used for the migration of virtualised services in an attempt to enhance service delivery within each peripheral network. Y-Comm's investigation of proactive handovers and mobility prediction are also two essential parts for the investigation presented in this thesis. The Y-Comm framework is presented in Fig. 3.7 and its constituent layers are explained below.

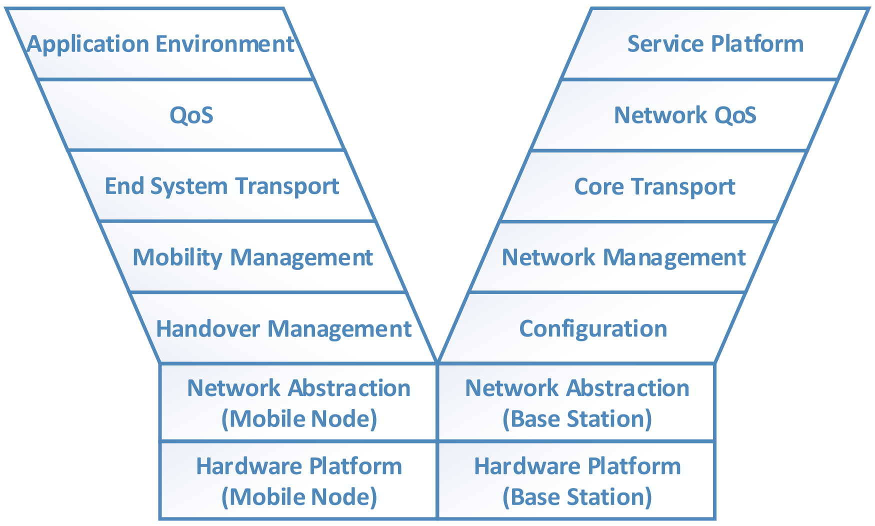


Figure 3.7 Y-Comm framework layers.

3.7.1 The Peripheral Framework

The Hardware Platform: *This layer defines the hardware network interfaces and underlying access technologies for each medium such as Wireless Interfaces and Media Access Control protocols.*

The Network Abstraction: *In this layer Y-Comm specifies a common interface protocol which must be supported by all networks. As the name suggests, this layer abstracts the underlying hardware from the upper layers and presents a single interface to the layers above for all types of networks.*

Handover Management and Mobility Management: *Mechanisms controlling vertical handovers are defined in this layer. Y-Comm defines two types of vertical handover. The network-controlled handover lets the network decide when a handover should occur. The client-controlled handover makes the client device responsible for controlling handovers.*

The Mobility Management layer contains policies that define when a handover should occur and instructs the Handover Management layer to perform a handover.

End-System Transport: *This is effectively the end-to-end transport layer of Y-Comm. Transport protocols that take into account the current state of network connection are defined to improve the performance of connections.*

Quality of Service: *This layer constantly reads the QoS parameters required by applications as well as the QoS offered by network connections and reports it to the layers below it so that the most appropriate networks for a handover are selected.*

Application Environment: *In this layer, Y-Comm specifies interfaces and mechanisms that enable applications to use the layers below.*

3.7.2 The Core Framework

Hardware Platform and Network Abstraction: *These two layers are similar to the layers in the peripheral network with the difference that they are on the network side rather than the mobile node. Hence, the protocols and mechanisms in the Network Abstraction layer are in this case software that runs on base stations rather than device drivers.*

Configuration: *This layer controls the configuration of Core network elements such as routers and switches and is tasked with reconfiguring networks in the optimum way in order to maximise efficiency and performance. For example, this layer contains mechanisms that dynamically and proactively allocate network resources before a handover occurs.*

Network Management: *The Network Management layer is the control place of the Core network, enabling administrative control of different domains through a common interface. Access control, accounting and charging systems are included in this layer thus making this layer responsible for presenting what resources are available for a handover to the Mobility Management layer in the Peripheral framework.*

Core Transport: *This layer defines transport protocols to be used in the Core network. Because Y-Comm assumes fast and reliable connections on the Core network, TCP is considered a sufficient mechanism for the Core network and therefore it is differentiated from the transport protocols to be used in the Peripheral networks.*

Network QoS: *Network QoS looks at QoS problems within the Core network while interfacing with the peripheral QoS layer to receive information from the client side. It passes this information to the layers below in order to optimise QoS for each connection.*

Service Platform: *The service platform provides an interface for services deployed in the Core network and their agents in order to use the layers below. One of the main functions of this layer is to provision services targeted to specific segments of the Internet such as regional services. This segmentation can offer QoS and security benefits.*

3.7.3 Evolution of Core and Wireless Networks

When Y-Comm was originally conceived, the Internet was still a heavily layered topology with clear boundaries between the different tiers of networks. As discussed in the previous chapter, the Internet has now evolved to the point where the differences between tiers are blurred due to the multitude of peering connections. As a result, the Y-Comm model of the Internet is now becoming relevant as the backbone of the Internet is becoming less layered and more of a mesh structure of high-speed connections. Multiple Wireless networks are now available in urban areas with each one connecting to Autonomous Systems that are part of the Core. There is a very clear separation between Core and Peripheral networks at present with the Core of the Internet using technologies such as Gigabit Ethernet and Fibre Channel, while the Periphery uses technologies such 3G, LTE and Wi-Fi.

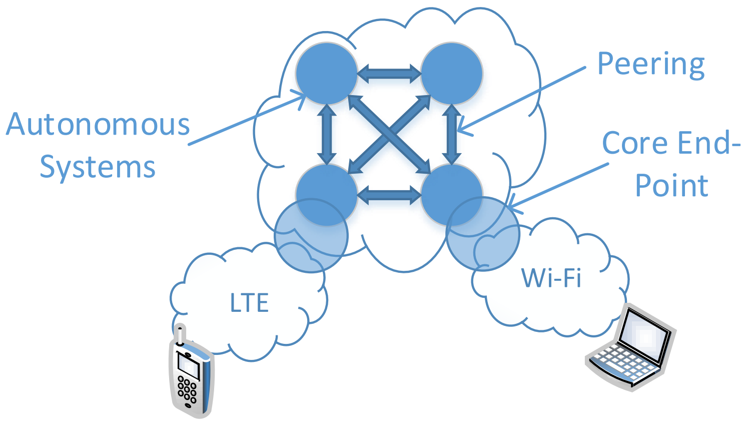


Figure 3.8 Envisioned Y-Comm Internet topology.

Fig. 3.8 shows how the Y-Comm model can now be related to the structure of the Internet. With this in mind, we can now look at how present day technologies and protocols fit in to framework and how they provide some of the mechanisms necessary to realise Y-Comm.

Current technologies such as SND and NFV discussed previously are currently in the phase of implementation and experimentation. These technologies along with new transport protocols such as the Simple Protocol (SP), proposed by Riley and Mapp, 2012 [56] and Multipath TCP, proposed in Ford et al. 2013 [21], are enabling the implementation of Y-Comm's theoretical framework in present day networks. Similar to how the TCP/IP model is the implementation of OSI, we will now look at the implementation model of Y-Comm with existing technologies. Before explaining the model, we will briefly look at SP and explain how it works and how it fits in to Y-Comm.

SP is a transport layer protocol that can provide reliable while running over unreliable protocols such as UDP or Ethernet. It does this by using control messages that define if and when reliability is needed. The structure of SP is shown in Fig. 3.9. The driving force for SP was the divergent path taken between technologies in Peripheral and Core networks. Peripheral networks are primarily using wireless technologies that are slower and more unreliable than their wired counterparts which are reaching speeds of 1Gbps and will soon reach 10Gbps. In the Core networks, connections are also reliable and use fibre that can reach 10Gbps. The developers of SP argue that while TCP is sufficient for the Core networks, the vast differences between wired and wireless connections in the Peripheral networks require a more flexible protocol in order to optimise performance and efficiency. SP presents a transport solution for LAN environments that is simpler and more flexible than TCP, thus optimising the performance of wireless and wired networks that a user is directly attached to.

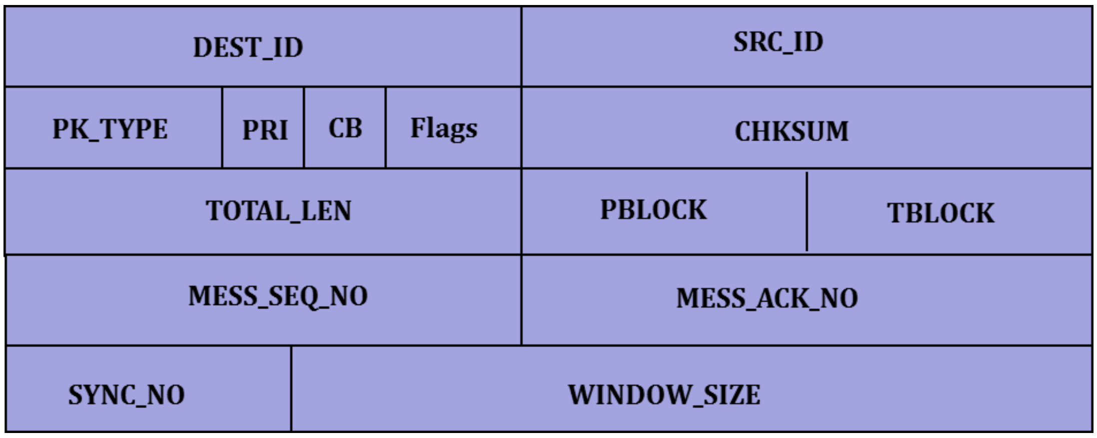


Figure 3.9 Structure of SP.

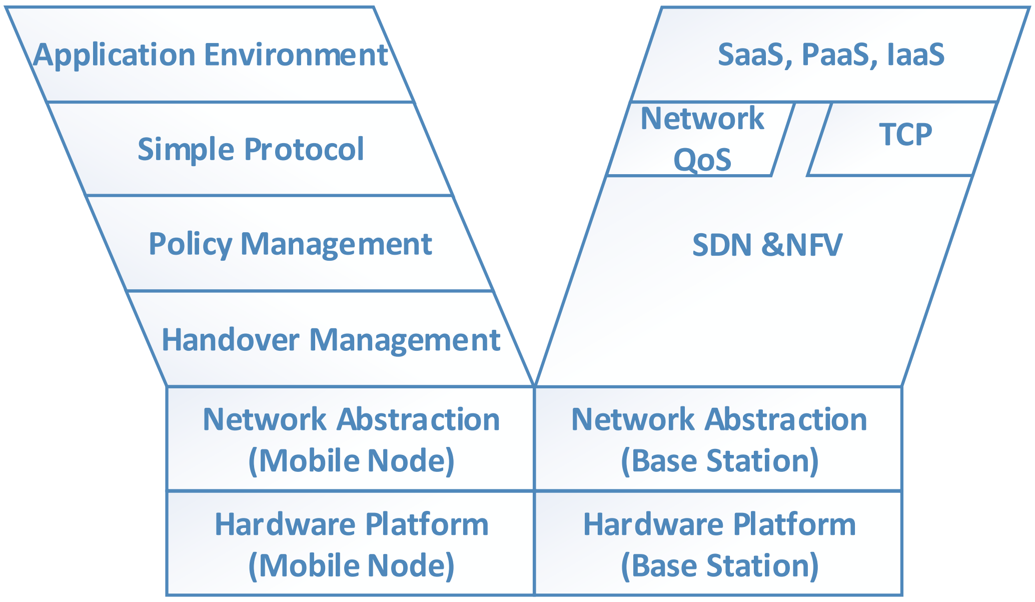

Application Environment
Simple Protocol
Policy Management
Handover Management
Network Abstraction (Mobile Node)
Hardware Platform (Mobile Node)
SaaS, PaaS, IaaS
Network QoS
TCP
SDN &NFV
Network Abstraction (Base Station)
Hardware Platform (Base Station)

Figure 3.10 Y-Comm implementation model.

The implementation model of Y-Comm, presented in Fig. 3.10, shows where SP is implemented for transport purposes. Owing to SP's ability to adjust reliability and QoS requirements, it is in position to send messages to the Mobility Management layer in order to find the best possible solution for a handover and thus optimise a connection in terms of the required QoS. In practice this means that SP can inform the Mobility Management layer about the active connections and their QoS requirements so that handovers to a more reliable or sufficiently reliable but less costly network can take place. In effect, this implementation enables client-based handovers for Y-Comm.

On the Core network side, we see that SND and NFV are taking the role of Network Management and configuration layers. As discussed in the previous section, SDN has the ability to dynamically configure networks in order to allocate resources where they are most needed. It can receive such information by monitoring QoS parameters in the Core network where TCP is primarily used as the transport protocol. By putting the two sides of the model together, we are now in position to see how network management and QoS can be monitored and adjusted for Peripheral and Core networks in real-time using these existing mechanisms. To better understand this scenario we envision an example of a mobile user that is moving away from the coverage of their LTE network and enters a Wi-Fi network. SP running on the mobile node will evaluate the required QoS parameters of active connections and the Mobility Management layer will decide if a vertical handover is desirable. If we assume that the Wi-Fi network offers good connection characteristics, the mobile node will disconnect from LTE and connect to the Wi-Fi. Concurrently to this process, on the Core side, SDN will detect that a new client has connected to the Wi-Fi and attempt to allocate extra WAN connectivity resources in order to keep the mobile node's traffic demands satisfied without hindering the performance of other Wi-Fi clients. The reverse of this process can take place when the client leaves the Wi-Fi coverage and reconnects to LTE. In conclusion, we see that current technologies combined with Y-Comm can improve the utilisation of resources for Peripheral and Core networks alike.

3.8 Mobility and Application Usage Patterns

To study ETM and network performance in the context of mobility, we need to have a good understanding of how users behave in their daily routine in terms of mobility patterns and application usage. Gonzalez et al. (2008) [22] state, that humans exhibit significant regularity in their mobility patterns because they visit frequent locations such as home and work. For each individual, it is possible to identify movement patterns in everyday life and although they are not unique to the individual, it is still hard to classify users into groups in terms of their mobility patterns. This classification becomes even harder when considering application usage patterns on mobile devices. For example we may be able to identify large groups of users that move every morning from north London to central London, however they do not all go to the same location and thus we cannot deduce any meaningful granularity in terms of who will connect to each network available in a large area. Furthermore, it is not possible to classify two users based on their mobility pattern because even if we assume that two individuals start at the same location and follow the same path to a single destination, their application usage patterns may differ significantly. The biggest problem, as described by Falaki et al. (2010) [18], is that user diversity can vary by orders of magnitude between individuals and thus it is not possible to cluster them in groups. Falaki et al. state that *"The diversity among users that we find stems from the fact that users use their Smart Phones for different purposes and with different frequencies. For instance, users that use games and maps applications more often tend to have longer interactions. Our study also shows that demographic information can be an unreliable predictor of user behaviour, and usage diversity exists even when the underlying device is identical, as is the case for one of our datasets."* In regards to traffic consumption, the effect of the above is reflected in the discovery that traffic sent and received across users differs by almost three orders of magnitude. Moreover, the volume of traffic exchanged that is considered interactive dominates by one order of magnitude the volume that is delay-tolerant. For approximately 90% of the users, over 50% of the traffic is interactive but for the rest, almost none of it is interactive. We also see that most users have a strong diurnal behaviour with 80% of them consuming over twice the amount of traffic during their peak hour. What this means, is that pursuing a classification scheme for mobile users that categorises them based on mobility and usage patterns is not an effective solution. Users should be treated as individuals and any traffic and QoS optimisations should occur for the individual. The consequence of this is that a centrally managed scheme is not going to be scalable or efficient and instead, a client-centric scheme may provide a better solution to QoS and traffic management.

3.9 Critical Summary and Insights

The efficiency of Cloud technology owed to its elastic management of resources has made it an industry-standard solution for providing services and virtualised development platforms. Entire infrastructures can be virtualised and various types of deployment allow tailor-made implementations of Clouds to support the needs of businesses and individuals. Perhaps the most interesting type of Cloud deployment is the Hybrid-Cloud where a Private Cloud built and managed by an entity can request extra resources when necessary from a Public Cloud that is built and managed by a different entity. However, different Cloud platforms are not able to interoperate due to lack of standards or loose adherence to them. To resolve this, Cloud interoperability standards are now in development and testbed platforms are being implemented that will lead to well defined standards for virtualisation technology and enable interoperability of Clouds.

Clouds are not restricted to providing virtualised applications for end-users and the technology is now used for Network Function Virtualisation (NFV) that until recently required specialised, purpose-build and expensive hardware. Network operators can now build their own datacentres with commercial off-the-shelf hardware and virtualise parts of their networks thus driving down their CAPEX and OPEX. This desire to drive the CAPEX and OPEX lower, has led to the development of technology which decouples the management plane from the hardware and allows for a more efficient and dynamic allocation of network resources. Equipment that supports SDN functions is controlled by a centralised entity that has an overview of network operations. QoS and routing policies defined by administrators are entered in the management place which then configures the network in real-time in response to traffic demands and QoS changes. The outcome is a more efficient utilisation of network resources, without the need for overprovisioning bandwidth in anticipation of high bandwidth demand.

Mobile Cloud Computing provides a platform for augmenting the capabilities of mobile devices through the use of Cloud resources available on the Internet. This is achieved by offloading parts of applications from the mobile devices to the Cloud. Perhaps the most complete solution provided by MCC is the complete virtualisation of mobile applications and the conversion of mobile devices to thin clients. In this case, the performance of the network plays a prominent role to the responsiveness of the applications and the overall user experience. For this reason, the localisation of Clouds is preferred as it has the potential of enhancing the performance of networks and keeping traffic contained within Autonomous Systems.

5th Generation networks will provide constant and reliable connectivity to mobile clients, thus setting the foundation for converting mobile devices to thin-clients and centralising all the processing in the Cloud. 5G technologies provide a more fine-grained approach to traffic management and QoS provisioning and give us the opportunity to use these mechanisms for the dynamic localisation of MCC services depending on network conditions and user mobility. To achieve this, we need a new service delivery framework that encompasses the intrinsic characteristics of MCC, 5G networks and human mobility.

Taking into account the information from the previous chapter, we proceed to create a list of insights that will help us define some of the functional requirements for a Mobile Service Delivery framework based on Cloud services with Cloud Interoperability and new network technologies in mind.

Insight #1 Cloud computing is evolving to become a more open platform via interoperability mechanisms that allows providers to cooperate and form Cloud federations for load balancing and QoS purposes. Cloud-based services and Service-Oriented Networks can further enhance the QoS and perform load balancing more efficiently on a global scale.

Insight #2 The Internet is evolving into a true mesh topology with network operators striving to peer with each other as much as possible in order to improve the QoS for their clients and reduce their CAPEX and OPEX by eliminating costly transit connections. Technologies such as SDN and NFV along with application of ETM in some cases help accomplish these targets.

Insight #3 Mobile devices are relying on Clouds for extra processing and storage resources. However constant connectivity is a requirement and mobility issues need to be addressed. To enable truly mobile thin-clients we must first understand human mobility patters as well as application usage patterns. This way we can improve the QoS and QoE and address traffic congestion problems that arise from the constant connectivity that thin-clients demand.

Insight #4 Web caching and CDNs work well for static content that can be copied to multiple locations and sourced from there. These technologies work well for generic content that can be distributed to many people at once without making it personalised or with small elements of personalisation. A personal VM in an MCC environment is not something that can be cached and distributed from multiple locations simply because it is specific to its user and not for the general public.

Insight #5 Seamless vertical handovers for mobiles will ensure constant connectivity and facilitate a more extended use of Clouds and other online services to the point that mobile thin-clients may be plausible. However, vertical handovers imply a change of network provider and consequently, an unpredictable and constantly changing flow of traffic caused by user mobility. To achieve ETM and QoS improvements via traffic localisation we need a dynamic solution that can migrate services in real-time according to the user's location.

Insight #6 Although emerging network technologies focus on improved utilisation of resources and provide QoS guarantees, the fluctuation of traffic throughput at different times of the day and during different events can cause a violation of these guarantees. A device should be able to confirm independently and automatically, that an available network can provide the QoS their applications require, reliably and consistently.

Chapter 4 Service Delivery for Mobile Clients

Offloading tasks from mobile devices to the Cloud is a process that involves several parties. We have Cloud and Network Operators and Service providers responsible for setting up and delivering a service and on the client side we have the user and the mobile device. Taking into account the list of insights from the previous chapter, we proceed to define some of the functional requirements for a Mobile Service Delivery framework. This chapter presents the framework and describes the functionality of its layers. The framework is used as the guideline for moving this investigation forward.

4.1 Requirements

With the above insights in mind, we can build a set of requirements for a mobile service delivery framework that makes use of current technology trends and the state of art.

Requirement #1 A service must be identifiable by a unique ID and bound to a set of parameters that can interoperate with platform providers. Such parameters must include the minimum requirements in terms of CPU time, storage and memory, network bandwidth and latency, security protocols, and dependencies on other services.

Requirement #2 Services must allow their users to personalise them in terms of performance. Each user may choose if they desire a better level of performance from a service or extra features. Such parameters may include: allocated bandwidth to the user, maximum latency, amount of storage, security level, and other processing resources.

Requirement #3 Platform providers (Clouds) should be able to accept or reject services depending on their set of requirements. They should also be able to bill service providers for processing, network and storage usage for any services and components running on their Cloud. Additionally, it is up to the Cloud provider to decide which technology is going to be used for service migrations provided that it meets the service's minimum requirements.

Requirement #4 In order to provide maximum benefits to their users, services need to be aware of their QoS level on a per-client basis. A server should also be aware of the client's current location and network provider. Such data may be gathered directly by the service and its processes, the client's device or a transport protocol that can report such information. This information can be used to determine when and where to migrate.

Requirement #5 A service requesting migration must pass information about the client's network provider to the platform provider in order to find the best alternative Cloud to host the service. Preferably a Cloud that is directly peering or local to the client's network.

Requirement #6 Any Cloud offering its resources to incoming services must also be able to report nominal values of network bandwidth and latency to the user's network prefix. This is to ensure that an incoming service will not only have sufficient Cloud resources to run but also adequate network performance to deliver its content at the QoS requested by the client.

Requirement #7 Service clients should be able to select the best possible network for handover via a querying mechanism which will confirm that the desired QoS level is deliverable through the new network. In other words, clients will not rely on reported nominal values for determining the best network for their service. If a handover to a network with suboptimal QoS is imminent, the service should migrate to an appropriate location (if one exists) to improve the QoS.

4.2 Introducing the Framework

The proposed framework to addresses the above requirements consists of six layers that are presented in Fig. 4.1. The same figure also shows examples of data and mechanisms that map to each layer. Before proceeding to analyse the layers and how they correlate to other existing communication frameworks, it is important to stress that this is a theoretical framework and not an implementation framework. The layers represent the components needed to enable mobile service delivery but these components may be merged together or further divided upon implementation similar to how the OSI relates to TCP/IP.

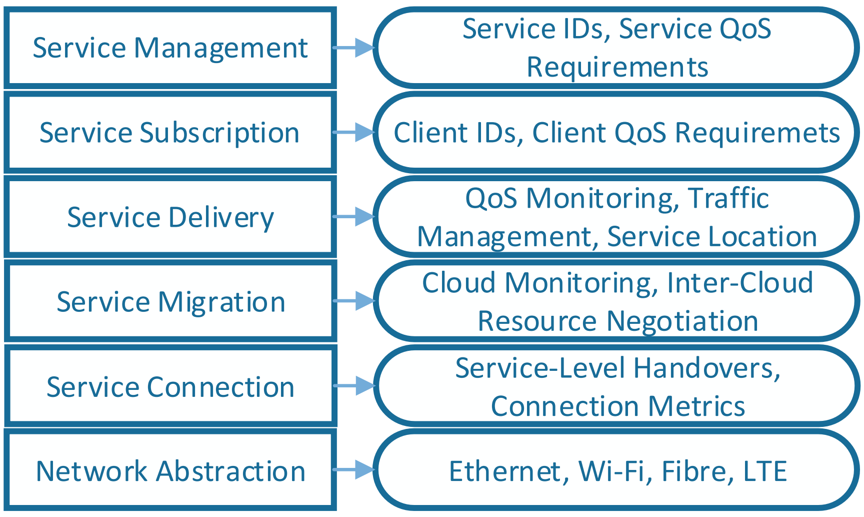

Service Management
Service IDs, Service QoS Requirements
Service Subscription
Client IDs, Client QoS Requiremets
Service Delivery
QoS Monitoring, Traffic Management, Service Location
Service Migration
Cloud Monitoring, Inter-Cloud Resource Negotiation
Service Connection
Service-Level Handovers, Connection Metrics
Network Abstraction
Ethernet, Wi-Fi, Fibre, LTE

Figure 4.1 Service delivery framework layers.

Service Management Layer: This layer deals with how services are deployed and managed. When a service is introduced to a service-oriented network or federated Cloud, it needs to have a unique Service ID so that it can be globally identified. Multiple instances of a service for load balancing or other purposes may also be identified by an Instance ID. The service provider must also define any dependencies to other services and access rights to those services if required. In this layer, the service provider must also define a set of minimum resources and requirements for the service to run. The amount of network bandwidth allocated, the maximum desirable latency, the number of virtual processors and memory size and the amount of persistent storage space are defined here. Finally, the provider must define security parameters for the service such as storage encryption and network encryption and they must also define if the service is allowed to migrate to third party Clouds and if so, under what conditions and requirements.

The set of data and requirements in this layer is what determines where a service may run and if a Cloud fulfils the requirements to host the service. Additional optional requirements may be added, such as preference to run on Clouds that use renewable energy resources or Clouds that provide resources for the smallest cost. In summary, this layer forms a Service Level Agreement (SLA) between a service provider and a Cloud provider and it implies the need for mechanisms that advertise the capabilities of Clouds and determine where a service may be hosted.

Service Subscription Layer: After successfully launching a service, clients from the general public may wish to customise the service to their needs. In this layer we create an SLA between the service provider and the client. Clients are identified via a unique Client ID which is also used for billing purposes and location tracking via mapping it to the user's network address. Each client is allowed to define a set of service parameters in their SLA that enhances their QoE compared to the minimum set by the service provider. For example the user may define in their SLA that they need additional storage or CPU resources. In the SLA, users may also define if they want to reveal their location to a service so that service migrations near their location may be enabled. This would require an additional set of parameters such as maximum desired latency which will then be used to determine when a migration should occur. Finally, additional security parameters above the minimum set by the service may be requested such as fully encrypting user content or restricting service migrations to Clouds with higher security levels than the minimum required by the service.

Service Delivery Layer: This is perhaps the most complicated layer as it brings together all the information provided be the layers above it and below it and determines how a service should behave in order to honour the SLAs of its clients and the provider. Various mechanisms are part of this layer. One example is a QoS monitoring mechanism where the network is queried for latency, jitter and throughput in order to determine if there are sufficient resources available to deliver the service to an acceptable level. Traffic management mechanisms also monitor traffic and determine if a service should move to a different location and where, in order to localise its traffic. Such mechanisms will need to query user devices for metrics and location in order to determine where the service should move to. Cloud monitoring mechanisms are also part of this layer and determine if a Cloud is providing the required resources. Once this layer decides how a service should be delivered and from where, it passes this information to the migration layer for inter-cloud negotiation of resources. The results of the negotiation are sent back to this layer for final approval before the service is moved to a new location.

Service Migration Layer: The service migration layer deals with the negotiation of resources between Clouds. The mechanisms in this layer reside on the Clouds and receive instructions from the service delivery layer. These instructions include the data from the service management layer as well as any extra conditions that are defined in the subscription layer. They also include the location of clients so that the migration layer can determine which Clouds are closest to the users of the service. Multiple negotiations may occur, resulting in multiple instances of the service created at different Clouds if necessary in order to cover a large population. The actual method of migration is also negotiated at this layer including network throughput guarantees to ensure that a service replicates within a specified timeframe. In case multiple Clouds fulfil the requirements for serving a specific user group, this layer may employ auction mechanisms to determine which Cloud should receive the service based on cost/performance parameters, or it may pass this information to the service delivery layer for the mechanisms within it to decide.

Service Connection Layer: This layer deals with how clients connect to the service and mainly contributes to the framework via transport mechanisms that can report the state of a connection to the Service Delivery Layer. These mechanisms are ideally implemented at the transport protocol but they could also be separate processes that gather data heuristically by monitoring traditional transport protocols such as TCP. It would not be scalable to implement such mechanisms at the service itself as it would have to actively gather data from all the clients on top of serving client request, so the ideal implementation would be for the clients to gather network metrics and report them to the service via small telemetry packets. This method complies with Y-Comm's idea of client-initiated network actions such as handovers and can also make use of Y-Comm's QoS monitoring mechanisms. Transport protocols such as the SP are capable of reporting such metrics for the purposes of this layer.

Network Abstraction Layer: As the name suggests, the main task of this layer is to mask the underlying network technology from the service and the clients. Once again, this layer follows the Y-Comm model of abstracting the underlying physical connections from the services and applications. The purpose of this abstraction is to allow the protocols at the connection layer to adapt to network conditions and gather performance metrics without technology-specific considerations.

4.3 Framework Implications

Starting from the last layer of the framework, we will use a bottom-up approach to understand its implications and how it could be implemented and operated. By abstracting the physical network from the transport protocols, we are effectively eliminating any compatibility or performance concerns that may present problems to mobile users and services. Similarly to service-oriented networks, this framework completely hides the underlying physical infrastructure from the services and focuses on how protocols can adapt to changing network conditions rather than adapting to specific technologies. For example, the applications or services do not have to worry about handovers between Wi-Fi and LTE and instead can focus on adapting to network QoS changes. From the perspective of the application, the physical network path is invisible and the only thing that changes is the characteristics of the connection when a handover occurs. The handover event itself is transparent to the service and when done successfully, there is no packet loss and need for data retransmission. The service communicates with its clients using Client IDs and Service Instance IDs and the micromanagement of network addressing schemes is left to the network. The only information that is passed to the upper layers is the user's current network prefix or ID so that the service can locate its users. All this, creates a fluid network landscape as shown in Fig. 4.2, where paths between services and clients are dynamic and it is the performance characteristics of these paths that determine the best route between services and clients rather than hop-counts and routing tables.

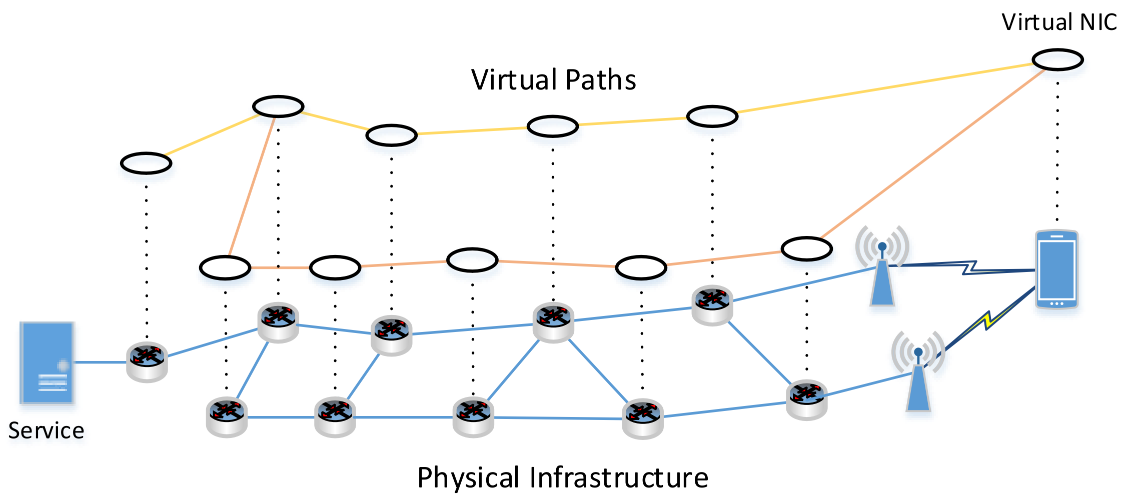


Figure 4.2 Dynamic path formation using the proposed framework and SOA.

The second implication stemming from this framework is that services need a transport mechanism that should be able to read the status of a connection and determine which network path is more appropriate for the desired QoS. They can then instruct the network to use the desired path if multiple options are available. This mechanism can either be implemented naively by querying the network for QoS metrics and therefore trusting nominal values advertised by a provider, or it can use a smart mechanism that probes network paths with dummy connections and then uses the gathered metrics to make a decision. There are arguments for and against each method. The main argument for the first approach is that it is simpler to implement. SDNs can report values such as load, latency and bandwidth but the gathered data may not be accurate enough to make a correct decision. Furthermore, a service may have varying requirements, so this method would demand from developers to determine various sets of requirements for each possible scenario that a service may encounter. We can understand this more clearly if we consider a remote desktop connection to a VM. The user may be processing a document and therefore, high bandwidth and low latency are not required or the user may be playing a game where latency and bandwidth requirements will depend on the type of game as explained in the previous chapter. It would be practically impossible to make a decision on which network path is best using theoretical requirements for the service. The second approach allows more flexibility because the service can probe the network with dummy connections that share the characteristics of its current connections. We can then make a more accurate decision based on the current state of the service and network paths. However, this means that dummy connections will consume some network resources while the service is probing and it also means that the process will take more time to complete. When considering a mobile device scenario (which is the focus of this framework), this means that probing has to occur before the handover (vertical or horizontal) to a new network so that the device will select the best possible connection (without considering wireless signal strength) in advance. In both cases, the goal is for the client device to connect not only to the network with the strongest wireless signal but also to the network with the best backhaul path to the service. From the service perspective, if this kind of selection fails on the client-side, it can use the same mechanism to determine a new location to move to.

The third implication of this framework is that service migrations have to occur within a specific time-frame. Since the user is mobile and the migration time also depends on how quickly the service is accessing and changing its working set, we need to determine the network resources required to move a service successfully. Since service migrations over WAN is a challenging problem we can safely assume that in a Federated Cloud environment, the participating Clouds will be peering with each other at private facilities to guarantee a certain level of QoS. However, this is not enough to guarantee adequate resources for migrations across Clouds, so we need a mechanism such as IntServ that will negotiate, allocate and guarantee network resources as and when needed for a migration to occur on time and in-line with the user's mobility and usage pattern. Factors that can determine the network resources required for a migration include the size of the service's working set, the rate at which pages are changed within the working set, how quickly the user is moving and the throughput of the user's connection to the service (for ETM purposes).

Another implication, coming from the top three layers of the framework is that service delivery mechanisms are aware of network location, mobility and connection characteristics on a per-user basis and they can use this data to determine the best way to deliver a service. This raises privacy concerns not only from the aspect of user location monitoring but also in terms of monitoring the actual usage patterns of a user as well as migrating their data freely across multiple Clouds that form a Federation. We can assume that Federated Clouds will have interoperability and security standards in place, however this does not mean that everyone should trust these standards or that everyone desires their private VM to move freely to third party Clouds. This problem is addressed at the top two layers of the framework, where a service provider or a client has the option to disallow migrations for their service or a personal service instance respectively.

Finally, the top two layers define the amount of resources that a VM or service requires to run while also allowing for individual clients to customise these requirements for their service instances. This fall within one of the Cloud requirements which is the ability to charge clients based on the resources they used for a period of time. What is implied in this framework is that a new market economy can be constructed where multiple Clouds may be competing for hosting services by advertising lower costs or higher performance. Users and service providers may prioritise cost or performance depending on the type of service and how they use it and the competition between

Cloud providers will offer more resources at lower cost for everyone. Clients will pay the service providers depending on the level of service they requested, and service providers will pay Cloud providers based on how many resources their services use.

4.4 Relating to Existing Communication Frameworks

In order to further examine and understand how this mobile service delivery framework operates, we will relate it to existing frameworks and relate some of its functions to their layers. We will start with the OSI model which is a theoretical model for communication. How the two frameworks are related is presented in Fig. 4.3.

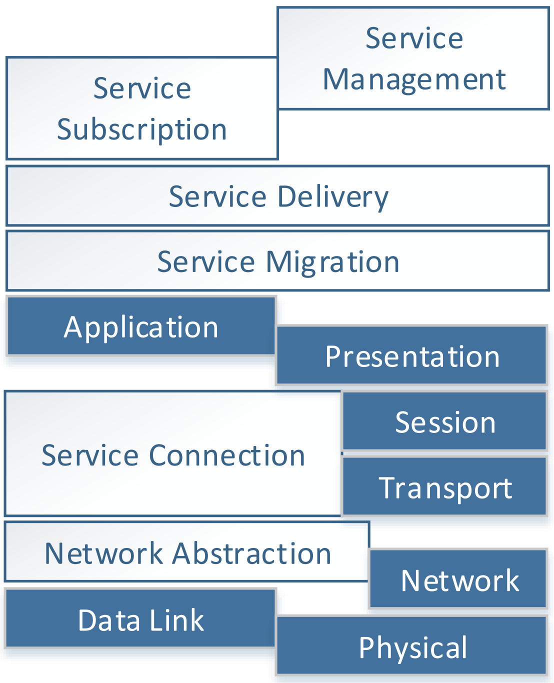


Figure 4.3 Mapping the proposed service delivery framework to OSI layers.

Starting from the bottom layers, we see that the network abstraction layer sits slightly above the network layer of the OSI. Network functions such as selecting an interface, establishing connectivity and acquiring an address are masked from the higher layers. However, for the purposes of link selection, the network layer may receive instructions from the higher layers via the abstraction layer. These instructions will typically include a set of performance parameters that will satisfy the client's application requirements. This is on top of any other link selection algorithms for wireless communication such as SNR-based binding. Due to this structure, it is up to the abstraction layer to resolve service and client IDs to network addresses and select the appropriate interface based on the requirements sent to it by the layers above.

Moving up to transport and session layers of the OSI, we relate them to the Service Connection layer of the framework. As mentioned previously, the main function of the Service Connection layer is to gather and report performance metrics of established connections so that the Service Delivery layer can decide how to best deliver the service. The rest of the layers of the framework sit hierarchically above the OSI since they are layers that manage the operation and delivery of applications, therefore we cannot directly map them to any of the OSI layers because the OSI does not provision for such functionality.

To put it all together in a practical example, we consider mobile node which is using a service running on a remote network. Let's assume the device to be connected to LTE and Wi-Fi concurrently and that the user is mobile and therefore the Wi-Fi connection may fail. In this scenario, the mobile service delivery framework combined with the OSI, will behave as follows: The Physical and Data Link layers will detect signal degradation at the Wi-Fi interface and switch all the communication to the LTE interface. When the Wi-Fi link fails, the Network Abstraction layer will report that a connection path is lost and advertise the QoS parameters of the LTE link. The Service connection layer will detect the network performance degradation and report it to the Service Delivery layer. The Service Delivery layer will have to compare the information from the Abstraction layer and the Connection layer to determine if the degradation is due to bad link conditions or bad backhaul path. At this point it may decide to move the service in a datacentre inside the LTE network or do nothing if the wireless signal is the cause of the degradation. Upon detection of new Wi-Fi networks, the abstraction layer will report their QoS parameters to the Service Delivery layer. At this point, dummy connections may be established to determine the backhaul capacity and QoS parameters. The best combination of backhaul capacity and signal strength will be selected. Once the connection is established and if the Wi-Fi network offers better overall QoS than the LTE, the Abstraction Layer will switch the flow of data to the Wi-Fi link.

In conclusion, the OSI was constructed as a communication model that does not account for user mobility or service-oriented networking. Consequently, many of the functions required for improving the performance of the future Internet are not directly supported by the model and have to be implemented by modifying or replacing existing mechanisms and protocols. New communication architectures that take into account user mobility and QoS can provide a better platform for the mobile service delivery framework.

To examine if this service delivery framework falls in line with modern communication architectures, we will examine how it relates to Y-Comm. We present how the framework layers correspond to Y-Comm in Fig. 4.4.

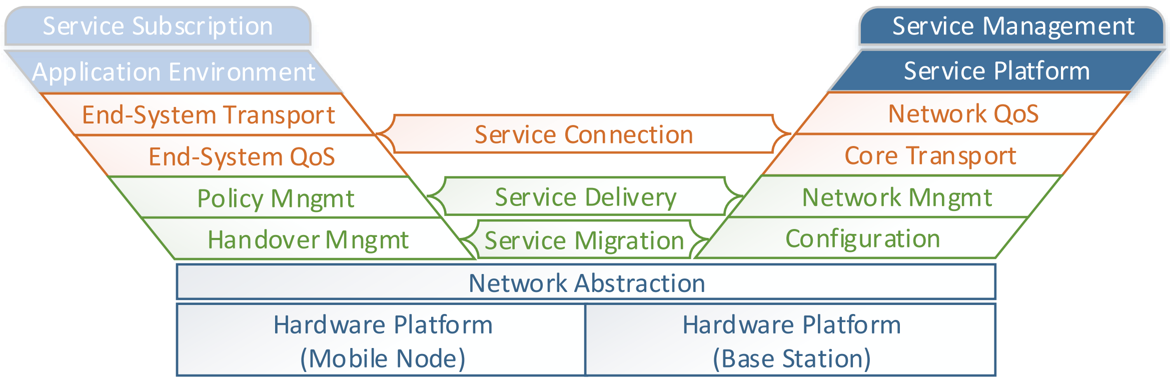


Figure 4.4 Mapping of service delivery framework layers to Y-Comm.

The first thing to notice when comparing the two frameworks is that a lot of the functionality expressed in the service delivery model, also exists in Y-Comm albeit at a different order. The Service Platform layer in the Core network defines the QoS parameters of services via Service Level Agreements (SLA). This directly relates to the Service Management layer of the service delivery model, where services define their QoS requirements via SLA. Similarly, on the Peripheral network, the Application Environment layer defines interfaces and mechanisms for applications to use the layers below it which has to do with setting QoS requirements and adapting to network changes. In this case, the Service Subscription layer defines these QoS requirements on a per-user basis and extends them not only to the network but also to the service itself, therefore providing a unified interface for the user to select additional network and service requirements. Therefore, when combining the top layers of Y-Comm and the proposed service delivery model, we see that they provide the parameters required for services to establish agreements with the infrastructure, clients to adjust their service preferences, and networks to auto-configure.

The Service Connection layer, in the case of Y-Comm appears higher than Service Delivery due to the way Y-Comm layers its functionality, however the tasks performed in the layer are unaltered. In the case of Y-Comm, the Transport and QoS layers of the Core and Peripheral networks gather network performance metrics and transport data. These metrics are passed on to the Management layers where decisions are made on how to route traffic more efficiently. Therefore the Management layers of Y-Comm directly relate to the Service Delivery layer of the framework. The biggest difference identified in the two frameworks comes from the fact that Y-Comm does not consider service migration and therefore lacks the functionality of dynamically moving services, however we can loosely relate the Service Migration layer to the Configuration layers of Y-Comm since these two layers are dealing with the negotiation and allocation of resources as instructed by the Management layer and it is therefore similar to how the Migration layer deals with the negotiation and allocation of resources for the migration of services as instructed by the Service Delivery layer.

In the case of Y-Comm we see that a modern communication framework has more similarities to the proposed Mobile Service Delivery framework. The need for QoS monitoring and dynamic allocation of resources is expressed in both frameworks and they also take into account factors such as per-user customisation of services and abstraction of the hardware from the layers above as a means of simplifying the functions of the higher layers. Thus, the proposed Service Delivery framework is best combined and implemented with Y-Comm because it can borrow many of its functions directly from the communication model without the need of implementing new mechanisms for QoS monitoring and network configuration.

4.5 Critical Summary and Research Focus

The framework presented in this chapter is a novel approach to optimising service delivery in the context of mobility using Cloud technology capabilities. As such, the framework takes into account the aspects of Cloud service delivery such as defining running parameters for a service and allowing a user to customise those parameters, while it also adds new aspects to service delivery that involve dynamic service localisation.

5G networks start taking into account mobility, especially in MCC scenarios and attempt to address some of the arising problems with mechanisms that dynamically adapt the networks to traffic conditions. This makes up for the lack of such mechanisms in current communication models; however, there is still no direct consideration about how services are delivered in a mobile environment and how services can be configured so that their performance will be optimised. The proposed framework combines network optimisation along with Cloud service optimisation by considering the needs of both and dynamically deciding how services will be delivered.

Each layer of this framework opens new areas of research both in terms of theoretical requirements and in terms of implementation mechanisms. In the following chapters, the focus will be on the Service Delivery layer which contains the mechanisms for determining when and where to move services for improved QoS and ETM.

Chapter 5 Traffic Management

This chapter focuses on the Service Delivery layer of the framework and here we explore the traffic management aspects for a scenario of a personal service or VM accessed by a mobile thin-client. The aim is to create a mechanism (reactive or proactive) that will dynamically move the service or VM based on the traffic being put through the network and the user's mobility patterns. A list of assumptions is presented in the following section.

Assumption #1: The scenario deals with a single service, accessed by a single client over the network.

Assumption #2: The service is running inside a VM which has a single VHD and access to the Internet.

Assumption #3: The VM has a "Home" network where the VHD is stored. If the VM has to contact its VHD over a third party network, then it is considered to be in a "Remote" network location.

Assumption #4: The client is mobile and the dwell time for the networks he visits is predicted and reported by the network or the mobile device.

Assumption #5: Networks to which the client is already connected or will connect in the future, are expected to have a datacentre that accepts incoming VM migrations on behalf of their clients for the purpose of localising traffic and improving QoS as shown in Fig. 5.1.

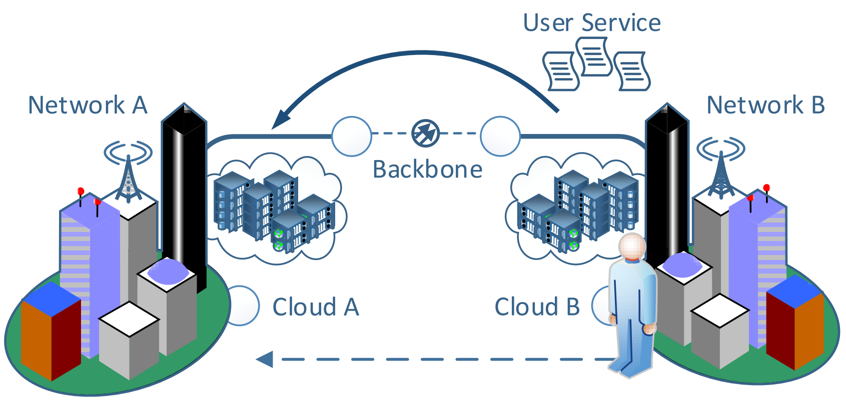


Figure 5.1 Assumed topology for dynamic service localisation.

5.1 Mobility and Network Dwell Time

The Y-Comm framework offers a solution for predicting user mobility alongside network connectivity and can contribute in making decisions on when to offload tasks and where. Because Y-Comm is primarily concerned with achieving seamless handovers across heterogeneous networks, an important trait of the framework is its ability to detect physical user mobility and consequently map it to potential network mobility based on which networks the user is likely to pass through. Mapp et al. (2012) [44] show how the Network Dwell Time (NDT) can be calculated for a single user by applying the Laws of Cosine in addition to knowing the user's location and velocity as well as the coverage areas of networks. They show that NDT in units of time can be estimated for a given speed and direction and network area coverage. Furthermore, the NDT for future network locations can be predicted based on the same input as illustrated in Fig. 5.2. For Y-Comm, the NDT is used to determine if a network is a potential target for handover. This information can also be picked up at the application layer and utilised for making applications aware of the user's mobility and network connectivity patterns.

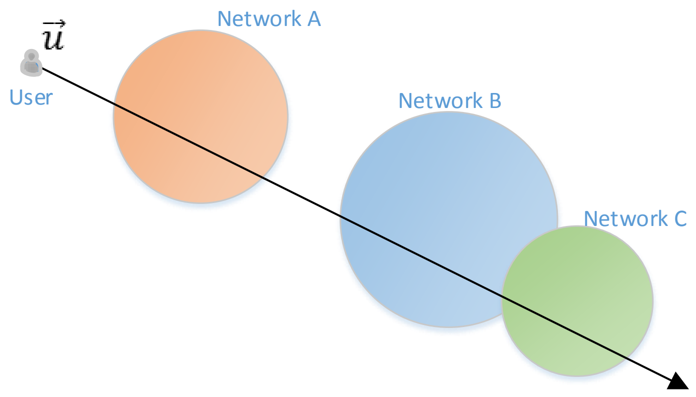


Figure 5.2 NDT and network connectivity prediction using Y-Comm.

Passing this information to the Application Layer, along with the network location of the user can facilitate the position of mobile services in a reactive or proactive manner. For example, a user that is about to enter network B, coming from network A, may proactively trigger a migration of his Cloud services to a datacentre within network B. What this means, is that given enough NDT and for specific traffic patterns, promoting the locality of these services can lead to an overall reduction on Inter-AS traffic. It may also lead to performance benefits given that hop count will be reduced. The essence of scheme is to apply ETM to mobile thin clients and their Cloud-supported services. Although ETM primarily deals with how mass traffic can be localised to minimise operator costs, using NDT we can adopt ETM for the benefit of mobile users and network operators.

5.2 Mathematical Analysis

Based on the assumptions above, two scenarios emerge for the migration of VMs. The first scenario is simpler and considers a VM moving from its home location to a remote location where the user currently is located. The second scenario, considers a VM that is already in a remote location and the user has moved on to another remote network location.

5.2.1 Scenario A

We will start by analysing the amount of traffic put through the network for a given NDT in the case where the VM is at the home location and the user just moved to a remote network as shown in Fig. 5.3.

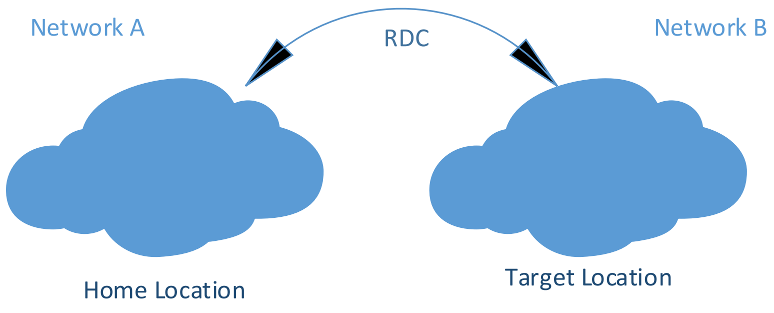


Figure 5.3 Simple scenario representation based on two locations.

In this case the total data that travels over the network is going to be:

$$Total\ bits = R \times t_{NDT} \tag{5.1}$$

Where R is the throughput of the RDC to the VM and t_{NDT} is the user's predicted NDT at their new location. Since the bulk of the RDC connection is going to be inbound to the user, network providers at home and remote locations will have an incentive to move that VM to minimise the amount of traffic that exits/enters their network. Moving the VM to the remote location is going to put data over the network, equal to the size of the

VM in addition to the RDC connection's data while the migration is underway. After the migration completes, the RDC data will be localised but the VHD data will have to be exchanged between the home and remote locations for the remainder of the user's NDT. This gives us the following equation:

$$Migration\ cost = C + (R \times t_{mig}) + D \times (\ t_{NDT} - t_{mig}) \tag{5.2}$$

Where, C is the capacity of the VM's RAM, D is the throughput of the VHD and t_{mig} is the time to migrate the VM. To calculate the VM migration time, we divide the size of the VM by the estimated throughput of the migration as shown below:

$$t_{mig} = \frac{C}{B} \tag{5.3}$$

Where, B is the throughput of the backbone link that will carry the migration traffic between the two networks. This throughput can be reported by a protocol such as the SP or it can be agreed upon in advance by the involved parties either with private peering and guaranteed QoS agreement between the two datacentres or using SDN to configure the networks so that a channel with guaranteed QoS is created for the duration of the migration.

We can now combine the equations (5.1), (5.2) and create the following expression:

$$R \times t_{NDT} > C + (R \times t_{mig}) + D \times (\ t_{NDT} - t_{mig}) \tag{5.4}$$

This tells us that if the total amount of bits expected to cross AS boundaries without migrating the VM is greater than the total amount of bits of the migration and subsequent VHD traffic, then it will be beneficial to move the VM to the remote location. Consequently, by sampling the RDC and VHD traffic of the VM we can generate values that we can put in the equation and along with the NDT reported by the network, we are in position to know when to migrate a VM.

5.2.2 Scenario B

In the second scenario, three networks are involved: Home, VM and User. This scenario occurs after a VM has already migrated to a remote network but the user has moved again to a new network away from home. The diagram demonstrating this scenario and the traffic flows involved is presented in Fig. 5.4.

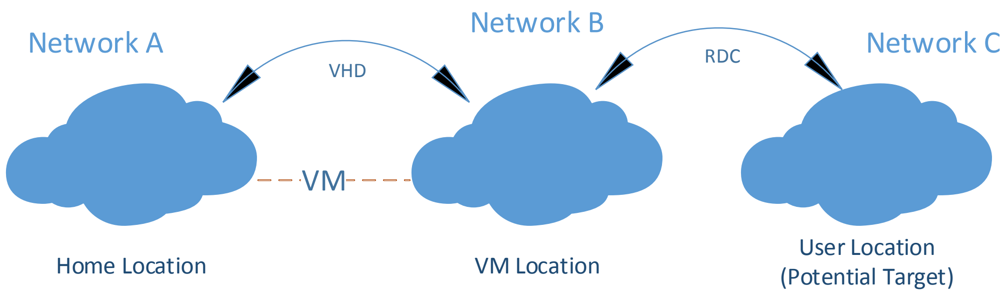


Figure 5.4 Service localisation using multiple locations.

We see that in this case we have three networks directly involved with delivering the service to the client (without counting transit or peering networks between them). In this case, the above equations are slightly modified depending on how the traffic is balanced between RDC and VHD connections.

When the RDC traffic throughput is equal to the VHD, then we have a choice of eliminating either the RDC traffic by moving the VM to the new target network, or the VHD traffic by moving the VM back to the home location. Because in this case the traffic is balanced, it would be more beneficial to eliminate the VHD traffic and default the VM to the home location, thus releasing resources from remote Clouds that would have otherwise hosted the VM. Therefore, the more efficient overall solution in this case, is to move the VM to the home location and only exchange RDC traffic across the networks. Therefore, if the total amount of inter-AS traffic for the predicted NDT is larger than the size of the VM, plus the RDC and VHD throughput during the migration, plus the RDC traffic for the remainder of NDT after migration, then it will be beneficial to move the VM back home. The following equation describes this mathematically:

$$t_{NDT} \times (D + R) > C + (R + D) \times t_{mig} + R \times \left(t_{NDT} - t_{mig}\right) \tag{5.5}$$

This can be simplified as follows:

$$t_{NDT} \times D > C + D \times t_{mig}$$

(5.6)

When the RDC throughput is greater than the VHD, then it is beneficial to eliminate the RDC traffic from crossing inter-AS boundaries and therefore, the aim is to migrate the VM to the user's new location. This leaves the VHD connection as the only inter-AS traffic after the migration occurs. The balance equation in this case becomes:

$$t_{NDT} \times (D + R) > C + (D + R) \times t_{mig} + D \times \left(t_{NDT} - t_{mig}\right)$$

(5.7)

This is simplified as follows:

$$t_{NDT} \times R > C + R \times t_{mig}$$

(5.8)

Finally, when the VHD throughput is greater than the RDC, similar to the first case, we try to eliminate the VHD traffic by defaulting the VM to the home location and therefore we are effectively reusing equations (5.5) and (5.6).

5.3 Flow Chart

Before attempting to put the above scenarios in a flow chart that represents all the possible migration outcomes, we must also consider proactive migrations as a result of the network reporting multiple NDTs as a result of a user passing through multiple networks in their path. For example, if the network is capable of predicting a user's path and which networks they will encounter in that path, it may also be possible to return an NDT for each network that the user is likely to join. In such an event, we may use solve the equations for NDT and migrate the VM to the first network in the user's path that has sufficient NDT to warrant a migration. However, if it happens that there is insufficient NDT in any of the forthcoming networks, we can calculate the total NDT and find out for how long the user will be away from the current location of the VM. This proves to be useful when the VM is moving away from the home location, as the total NDT can be used to calculate the amount of data crossing AS boundaries as a result of the VHD connection of the VM. Since we can find no firm target for a migration and therefore it is inevitable for the RDC traffic to cross AS boundaries, we instead choose to eliminate VHD traffic based on the total NDT of the user. We use the total NDT for all the networks as an input in (5.6) and we calculate if there will be traffic savings by returning the VM to the home location. This has the added benefit of releasing Cloud resources from the intermediary running the VM and only putting the RDC traffic through the network as opposed to having VHD and RDC traffic crossing AS boundaries.

One notable quality of the above is that this particular method of pursuing traffic localisation does not restrict itself to optimising traffic between the two or three networks involved in delivering the service. Instead, it provides a more general solution that minimises inter-AS traffic based on a user's predicted mobility patterns and also releases Cloud resources from networks that need not be involved in delivering the service. With this in mind, the flow chart for all the possible scenarios of the migration process is as shown in Fig. 5.5.

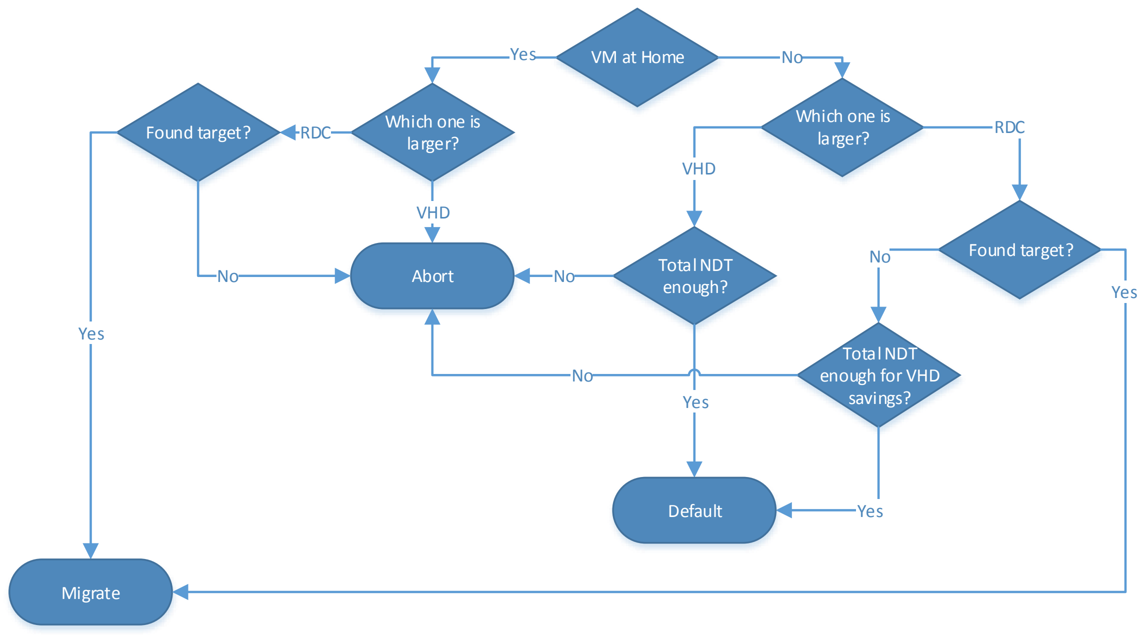


Figure 5.5 Migration process flow diagram.

5.4 Prototype

As a proof of concept, the above equations were used in a PowerShell script that monitors a VM's RDC and VHD throughput and automatically moves the VM based on the results of the equations. The NDT and migration throughput are given by the user when the script is executed so that the calculations can be performed. The methodology and experimental platform are described in the following section.

5.4.1 Prototype Platform Specifications

A basic virtualisation platform consisting of two physical nodes in a domain configuration was used as the basis of the experiment. Details of the two nodes are as follows:

Table 5.1 Prototype platform host specifications.

Host A	Host B
Intel Core i7 920, 16GB RAM	Intel Core2Quad Q6600, 4GB RAM
Kingston HyperX SSD 128GB	Crucial C300 SSD 128GB
Windows Server 2012 R2 Enterprise x64	Windows Server 2012 R2 Enterprise x64
2x Gigabit NIC on PCIe Bus	Gigabit NIC on PCIe Bus
	100Mb/s NIC on PCIe Bus

Both nodes are configured with one Gigabit NIC connected to a gigabit switch acting as the backhaul connection for VM migrations and for server management. The remaining NIC of each node is connected to a 100Mb/s switch, acting as the front-end network where the client connects. The 100Mb/s switch has a built-in wireless access point which is used for connecting the client.

A virtual machine, acting as the Domain Controller was set up on Host A which has more RAM and CPU resources available. The Client VM along with its VHD is also initially set up on Host A which acts as the home location for the VM. This means that the Home location for both VMs is Host A since it is the host that holds the VHD for each VM. The details of the two VMs are as follows:

Table 5.2 Prototype platform virtualisation configuration.

Domain Controller	**Client**
2 Virtual Cores, 1GB RAM (Dynamic)	4 Virtual Cores, 2GB RAM (Dynamic)
1 virtual NIC connected to the backhaul network	1 virtual NIC connected to the front-end network
20GB VHD (Dynamic) on Host A SSD	20GB VHD (Dynamic) on Host A SSD
Windows Server 2012 Enterprise x64	Windows 8.1 Professional x64

A third physical node was connected to the backhaul network for administrative purposes. A laptop acting as a thin client was connected to the front-end network wirelessly at 54Mb/s using 802.11g. The network diagram is shown in Fig. 5.6.

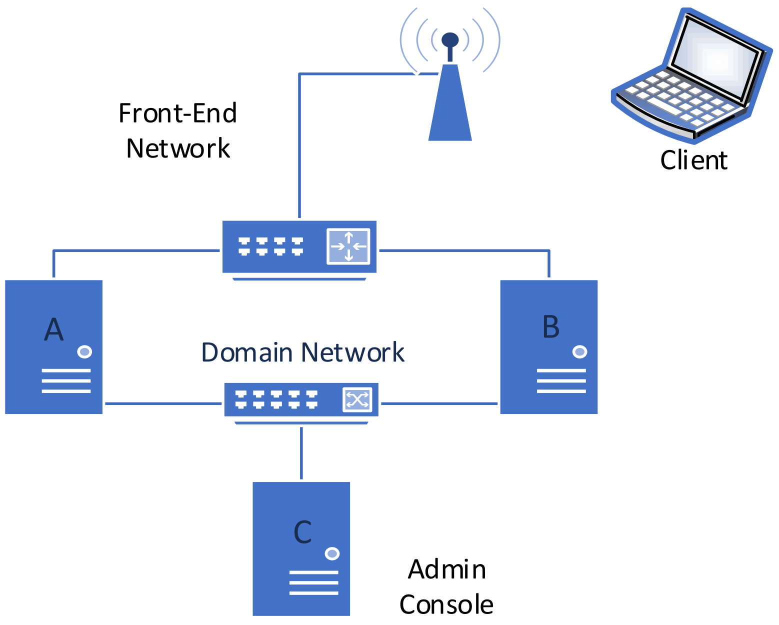


Figure 5.6 Prototype platform network diagram.

5.4.1.1 Preliminary Testing and Performance Measurements

A set of preliminary tests was carried out and presented by Sardis et al. (2014) [58] for the purpose of finding some base values to be used as input for the migration throughput. In these tests, the VM was moved along with its VHD between the two hosts and the results showed that the determining factor for the migration throughput was the SSDs on the hosts. The SSD read/write speed was acting as a bottleneck, preventing the full Gigabit bandwidth from being utilised. However, during the last step of the migration, when the RAM contents of the VM were copied, the bottleneck was the network throughput. The throughput value for moving the VHD was on average 50MB/s, while the throughput value for moving the VM only was fully saturating the Gigabit Ethernet link at 117MB/s. This was achieved on an idle VM. Further testing, showed that when the VM was in use (user playing a game), the throughput was on average 80MB/s due to the extra time it takes to copy RAM pages that are being modified by the application. This value was used as the base input for the estimated migration throughput in the final experiments. Extra tests to identify problems with migrations over lower capacity links showed that the most detrimental factor to the process is network latency. Any tests performed with latency over 50ms, caused the migration to abort. The base latency of the link between the nodes is less than 1ms and latency was shaped using Connection Emulator (Softperfect, 2014) [62]. Bandwidth had a linear effect to the migration duration and the process completed successfully at up to Fast Ethernet speeds as per Microsoft's documentation. Tackling WAN emulation problems is not within the scope of this experiment, so the full Gigabit capacity of the backhaul network was used without any tampering. Any changes to the QoS before the point where migration is impossible are reflected as a smaller throughput and therefore longer migration time. This is why it was initially stated that QoS guarantees should be in place in advance or a mechanism that dynamically probes the link capacity is needed.

5.4.2 Script Logic

Two PowerShell scripts were used to perform automatic migrations during the experiment. The Reactive script, takes as input only one NDT and assumes that the script will start running at the moment the user connects to a new network, or during the handover process between two networks. The logic process for the reactive script is as follows:

```
1.  Read the local hostname.
2.  Ask the user for the ServiceID (VM name).
3.  Query Hyper-V to find the current size of the VM's RAM.
4.  Ask the user for the estimated migration throughput.
5.  Calculate estimated migration time using VM size and migration throughput.
6.  Ask the user for the estimated NDT.
7.  If user NDT is less than the migration time, abort the script.
8.  If NDT is greater than migration time query Hyper-V to find the path to the VM's VHD and do a
    match check between hostname and path to find out if the VM is running on the same host
    where the VHD resides (Home location).
9.  If the VM is running at Home location, measure VHD throughput via the Hyper-V and RDC
    throughput via the front-end NIC.
    9.1. Repeat measurement 10 times and report the average throughput for RDC and VHD.
    9.2. If RDC is less or equal to VHD, abort the process.
    9.3. If RDC is greater than VHD, calculate the minimum NDT required for traffic savings
        9.3.1.If NDT is greater than NDT minimum, move the VM to target host.
        9.3.2.If NDT is less than NDT minimum, abort the process.
10. If VM is running at remote location, measure VHD throughput via the Backhaul NIC and RDC
    throughput via the Front-End NIC.
    10.1. Repeat measurements 10 times and report the average throughput for RDC and VHD.
    10.2. If RDC=VHD=0, abort the script (VM inactive).
    10.3. If RDC less than VHD calculate minimum NDT required for traffic savings.
        10.3.1.  If NDT greater than minimum NDT, default the VM to Home.
        10.3.2.  Else, abort.
    10.4. If RDC greater than VHD, calculate minimum NDT for traffic savings.
        10.4.1.  If NDT greater than minimum NDT, move the VM.
        10.4.2.  Else, abort.
```

The proactive script takes a set of networks and their NDT as input and predicts where the VM should be moved in advance. The networks and NDT for each one are given to the script in an array in the order in which the user will connect to them. A second array maps the networks from the first array to host nodes that can accept migrations. The proactive script logic is as follows:

```
1.  Read the local hostname.
2.  Read the array of networks and NDTs.
3.  Read the array of networks and target hosts.
4.  Calculate total NDT from the array of NDTs.
5.  Ask the user for ServiceID (VM name).
6.  Query Hyper-V to find the size of the VM.
7.  Ask the user for the estimated migration throughput.
8.  Calculate the estimated migration time.
9.  If the total NDT is less than the estimated migration time, abort the script.
10. Else, query Hyper-V to find the path to the VM's VHD and do a match check between hostname
    and patch to find out if the VM is running on the same host where the VHD resides.
    10.1.  If the VM is running at home location, measure VHD throughput via Hyper-V and
           RDC throughput via the front-end NIC.
        10.1.1. Repeat measurement 10 times and report the average throughput for RDC and VHD.
        10.1.2. If RDC is less or equal to VHD, abort the process.
        10.1.3. If RDC is greater than VHD, calculate the minimum NDT required for migration.
            10.1.3.1. Select the first network in the array with NDT greater or equal to the
                      minimum NDT.
            10.1.3.2. Resolve the network name to a target hostname.
            10.1.3.3. Move the VM to target host.
            10.1.3.4. If no network has NDT greater or equal to the minimum NDT, abort the
                      process.
    10.2.  If the VM is running at remote location, measure VHD throughput via the backhaul
           NIC and RDC via the front-end NIC.
        10.2.1. Repeat measurement 10 times and report the average throughput for RDC and VHD.
        10.2.2. If RDC=VHD=0, abort the script (VM inactive).
        10.2.3. If RDC greater than VHD, calculate NDT minimum for migrating VM to a target host.
            10.2.3.1. Select the first network in the array with NDT greater or equal to the
                      minimum NDT.
            10.2.3.2. Resolve the network name to a target hostname.
            10.2.3.3. Move the VM to the target host.
            10.2.3.4. If no network has NDT greater or equal to the minimum NDT, calculate new
                      minimum NDT based on VHD traffic for moving the VM home.
                10.2.3.4.1. If the new minimum NDT is less than the total NDT in the array of
                            networks, move the VM to the home location.
                10.2.3.4.2. Else, abort the script.
        10.2.4. If RDC less or equal to VHD and RDC greater than 0, calculate the minimum NDT for
                moving the VM home.
            10.2.4.1. If total NDT in network array is greater than minimum NDT, move the VM
                      home.
            10.2.4.2. Else, abort the script.
```

5.4.3 Testing Setup and Methodology

In order to evaluate the applicability of dynamic service localisation based on user mobility and ETM strategies, two sets of tests were conducted. In the first set of tests, the user launched a game on the VM while accessing it via the front-end network using RDC. A casual game was selected as a representative sample of what a user might play on a mobile device. Pinball FX (available on Windows Store) is a 3D pinball game that features colourful rapid-changing effects and requires quick reaction times from the user and therefore low latency on the network. The colourful and dynamic interface of the game makes it a good application for stressing the RDC session. The RDC resolution was set to Full HD (1920x1080) because it is a representative screen resolution of current technology Smart Phones.

For the first set of tests, the user launched the game and the PowerShell script was executed manually via the Admin node on Host A (home location). The same test was repeated with the VM residing on Host B (remote location). The game was then closed and the VM was allowed to idle while the same scripts were again used to evaluate how they would respond to an idling VM. No measurements were recorded during this test as it was only used to confirm that the scripts behave as expected when the VM is idle.

For the second set of tests, the user launches ATTO Disk Benchmark, as a means for creating VHD throughput to simulate an application with frequent disk access. The script first runs with the VM residing on Host A and the test is repeated with the VM running on Host B. The benchmark software is closed and the machine is allowed to idle for a minute. The user then opens a word processing application, browses to a plain text file and opens it. The script is executed again during this process. Finally, the user closes the word processor, launches a web browser and navigates to a news website. The script is executed again during this process. The tests are concluded with the user locking the VM and disconnecting from the RDC session.

The two sets of tests were designed to shift the balance of throughput between RDC and VHD connection in order to identify any problems in the logic of the scripts. They were also designed to be representative examples of what a user might do on a mobile device, ranging from playing a casual game to editing a text file and reading the news. The chosen resolution is quite common on modern mobile devices such as Smart Phones and tablets as well as laptops. To simplify the prototype setup and focus on investigating the functionality of the proposed solution, real user mobility is not addressed but instead, the client is always connected to the same type of network through one network provider and mobility is simulated by changing the NDT. In a more complex prototype setup, a client could connect via different access networks and network addressing mechanisms as proposed by Harney et al. (2007) [25] can be used to dynamically update the VM and client addresses. This, however, is outside the scope of ETM and this thesis does not focus on address-related network issues.

5.4.4 Test Results and Analysis

The results of the experiments are presented in the below along with the NDT values used in each case. The migration throughput was pre-set to 80Mb/s as explained in the previous section.

Table 5.3 Reactive script migration results.

VM at Home (Host A) - Reactive Script				
VM (MB)	**NDT (sec)**	**RDC (MB/s)**	**VHD (MB/s)**	**Result**
1858	700	4.440	0.600	Moved to user's network
2048	500	4.400	1.500	Aborted: Insufficient. NDT
1292	500	0.000	63.320	Aborted: VHD>RDC
VM at Remote (Host B) - Reactive Script				
VM (MB)	**NDT (sec)**	**RDC (MB/s)**	**VHD (MB/s)**	**Result**
2048	500	2.265	0.200	Aborted: Insufficient NDT
2048	1500	2.110	0.540	Moved to user's network
1924	500	0.000	5.460	Moved home

Table 5.3 of the results shows how the reactive script behaved under different scenarios. With the VM running on Host A, migration occurred only when the NDT was sufficient (700 sec) and the RDC traffic was greater than VHD. When the VHD traffic was greater than RDC or when the NDT was below the threshold, the operation was aborted. After the VM was moved to Host B, the gaming and disk benchmark experiment was run again and this time the VM was moved back home when the VHD traffic was greater

than the RDC traffic. When the RDC traffic was greater than the VHD, the script aborted the operation when there was no sufficient NDT for the migration. The migration completed successfully and the VM was moved to the user's target location when an NDT of 1500 seconds was given. There are two things to note in this set of results: Host B always generated less RDC traffic while gaming compared to Host A. The cause of this was identified to be the hardware of Host B. While gaming on the VM, Host A reported 45% CPU utilisation from the VM, however when the VM ran from Host B, the reported CPU utilisation was 95%. The number of virtual CPUs of the VM did not change between hosts and therefore this difference in utilisation is narrowed down to the processing power of the CPU. The CPU in Host B is from an older generation and the same processing load translates to higher CPU utilisation. This in turn resulted in the VM adapting its RDC connection to lower image quality that generates less traffic towards the client and hence the lower RDC throughput. The level of interaction with the game was not affected but the image quality was. One way to tackle this problem is to enable 3D graphics acceleration on the VM but this requires special hardware which was not available at the time of testing. The game is using software rendering performed on the CPU and therefore the CPU poses a performance bottleneck in this case. Moving the VM back to Host A, always resulted in better image quality and higher RDC throughput. The front-end NIC on Host B is ruled out as a potential bottleneck because even at 4.4MB/s throughput, the Fast Ethernet is only operating at 50% capacity.

Table 5.4 Pre-emptive script migration results.

VM at Home (Host A) - Pre-Emptive Script				
VM (MB)	**NDT (sec)**	**RDC (MB/s)**	**VHD (MB/s)**	**Result**
1538	Multiple, Total 359	4.500	0.260	Aborted: No target with sufficient NDT
1638	Multiple, Total 2299	4.540	0.010	Found target with sufficient NDT and moved the VM
1176	Multiple, Total 1399	0.000	25.840	Aborted: VHD>RDC.
VM at Remote (Host B) - Pre-Emptive Script				
VM (MB)	**NDT (sec)**	**RDC (MB/s)**	**VHD (MB/s)**	**Result**
1544	Multiple, Total 36	2.350	0.050	Aborted: No target found and insufficient Total NDT for moving Home

1544	Multiple, Total 1650	2.330	0.250	Moved the VM to target with sufficient NDT
1100	Multiple, Total 36	0.000	15.930	Aborted: Insufficient Total NDT
1070	Multiple, Total 1399	0.000	15.000	Moved Home

The pre-emptive script differentiates from the reactive by taking an array of NDTs and target networks. The calculations for the migrations are performed first by checking if a migration to a target network is possible. If no network in the array has enough NDT for migration, the pre-emtpive script sums up the total NDT for each network and then uses that as input to determine if there will by any traffic savings on the VHD side by moving the VM back to its home location if it is not already there. If a target network is found, the pre-emptive script also differentiates from the original by using a second array which resolves network names to nodes that can accept migrations. In the test setup this aspect of the script could not be fully tested as there were only two available nodes, however by setting the hostname of Host B into the array, the script correctly resolved the network name to the hostname and moved the VM successfully, thus proving that this approach can also work.

As shown in Table 5.4, during the gaming tests, the script successfully moved the VM when sufficient NDT was given in one of the networks in the array while it aborted the operation when none of the networks has enough NDT and the total NDT in the array was not enough to move the VM home and eliminate VHD traffic. During the disk benchmark tests, the focus is not on finding a target network but rather on eliminating VHD traffic from the network. As such, when the total NDT was sufficient the VM was returned home while in the opposite case, the operation was aborted.

5.5 Critical Summary

This chapter investigates how Economic Traffic Management rules for traffic localisation can be applied to personalised services for thin-client devices that use MCC technology. In the presented example of a VM accessed by a single client, this approach to ETM provides a mechanism that can reduce inter-AS traffic by eliminating the traffic flows with the highest required throughput and therefore induce the highest cost both in economic sense and in resource management. The weakness of this approach stems from the fact that user mobility can be unpredictable and therefore the estimated NDT may change at any time of the user changes their speed and direction of movement. The probability of a false prediction increases when a user is moving at walking speed and is free to go in any direction, therefore exiting or entering networks in an unpredictable manner. NDT prediction is more accurate when the user has a predefined path from which they cannot deviate such as when driving a car and have to follow traffic rules or when they are on a train or bus.

The main advantage of the proposed solution is that unlike other ETM proposals, it is more fine grained and considers traffic on a per-user basis as opposed to using user clusters or swarms and trying to predict the probability of localised data on a network and its peers. Furthermore, it is a novel solution to studying the viability of WAN migrations from an ETM perspective by taking into account factors such as the migration throughput, the size of the VM and the network throughput of a service.

The prototype demonstrates that ETM-based migrations are possible and with the use of modern technologies and transport protocols (SDN, SP), they can become a reality. The prototype uses NDT albeit without real user mobility. This simplifies the testing by eliminating handover problems and focusing on the traffic management aspect of the experiment. In the LAN environment, the handover process after the migration is a simple ARP update issued by the hypervisor. In a WAN environment, the process would be more complex involving DNS and router updates and possibly an IP address update. However these problems have already been identified and are investigated by researchers trying to provide solutions for WAN migrations. For example, Harney et al. (2007) [25] have proposed the use of Mobile IPv6 for rerouting packets to the VM after is has moved.

The reduction of inter-AS traffic is mathematically proven when certain conditions are met and the experimental results prove that it is possible to be implemented with the deployment of traffic monitoring, network reconfiguration and mobility prediction mechanisms. However, it should also be noted, that due to unpredictability in human behaviour, one of the weaknesses of this solution is that VM migrations may be triggered based on conditions that may change shortly after the decision is made.

Chapter 6 QoS Management

This chapter investigates QoS provisioning in the context of user mobility. A novel queuing model is presented, where network performance and user mobility are among the determining factors for the overall QoS that a mobile device will receive from networks along a user's path. The methodology for solving the queuing model is presented along with two examples of how it can be applied. Finally, the implications of this model are outlined along with some advantages and disadvantages.

6.1 Modelling Network Performance in the Context of Mobility

Due to the structural complexities of the Internet and the vast amount of diverse interconnections between Autonomous Systems, the performance of connections can vary depending on the amount of traffic, geographical distance and routing distance between clients and services. As discussed previously, network performance and more importantly latency is a determining factor to the QoE in an MCC scenario. 5G network technologies are responsible for providing constant connectivity to mobile devices, however, this means that as users move and their devices switch between networks, there is a potential for experiencing a greatly varying QoS.

One way to mitigate this is by dynamically localising MCC services in such manner that the user is always served by the closest datacentre, thus minimising latency. As explained in Chapter 2, one of the deployment models for MCC is to use Public Clouds as persistent storage and task offloading platforms while also using localised datacentres for enhancing the QoS when possible. As also explained in Chapter 3, the closest datacentre is not necessarily the best option for varying reasons that mainly revolve around the Internet's structure. Furthermore, there may be cases where the QoS delivered by a Public Cloud satisfies the user's application's demands and therefore a migration of services to a local datacentre may be redundant.

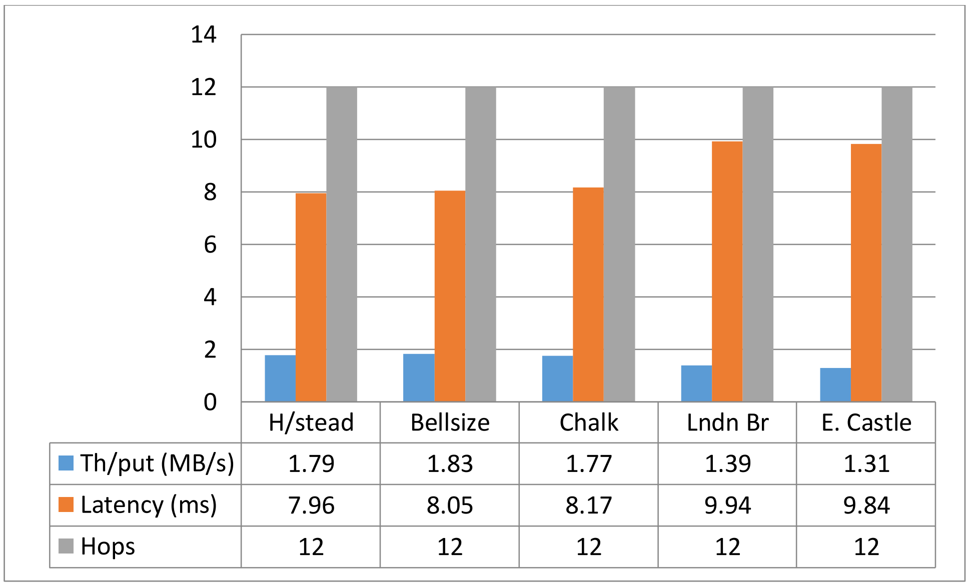
Therefore, the goal is to identify when a service should be localised based on QoS parameters akin to the method presented in the previous chapter for the purpose of traffic management. To achieve this, we can use a queueing model that takes into account the performance of various networks as well as user mobility and determines when service localisation is desirable in order to enhance the QoS. Before introducing the model, a set of experimental results is presented in the next section. The results were gathered by performing latency tests to a fixed server from various locations in order to determine the relationship between hop-counts, geographical distance and overall latency.

6.1.1 Geographical Distance and Latency

The latency tests were carried out using public Wi-Fi hotspots and mobile networks from Vodafone and EE. To access the mobile networks, a Smartphone was used in Wi-Fi hotspot mode and shared its mobile network connection with the client device. Measurements were taken from a laptop using the PsPing utility for latency measurements and traceroute for recording the routing path between the laptop and the server. The target server was installed at Middlesex University's Hendon Campus and connected to the Internet via the Joint Academic Network (JANET). The purpose of this experiment is to study end-to-end network performance rather than prefix-to-prefix or node-to-prefix. This way, we can look at the entire path of the connection and identify where performance bottlenecks are occurring.

6.1.1.1 London Tube Wi-Fi Latencies

The first test involved the client using the Wi-Fi network in London's tube service which is provided by Virgin. Starting from Hendon Central and moving south on the Northern Line, tests were conducted at each station that has free Wi-Fi available. PsPing was configured to perform TCP ping and measure the bandwidth. In total, 20,000 packets were used for latency measurements and 20,000 for bandwidth measurement. TCP ping was chosen because it is more representative of the connection between service and client compared to ICMP pings. Furthermore, the packet size was set to 1024 Bytes so as not to overwhelm the server and the network. Results are presented in Fig. 6.1 in ascending order based on geographical distance from the campus.



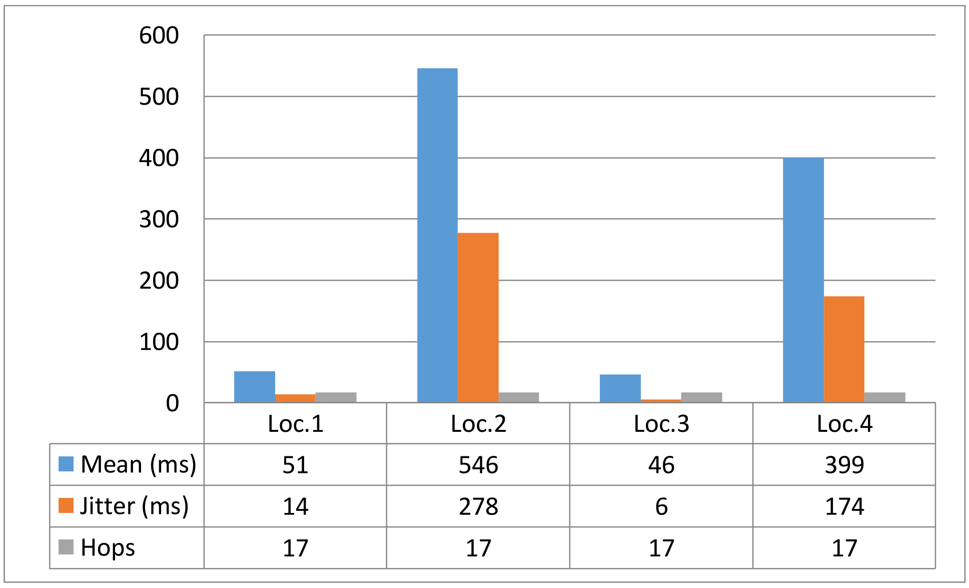
	H/stead	Bellsize	Chalk	Lndn Br	E. Castle
🟦 Th/put (MB/s)	1.79	1.83	1.77	1.39	1.31
🟧 Latency (ms)	7.96	8.05	8.17	9.94	9.84
⬜ Hops	12	12	12	12	12

Figure 6.1 London Tube results in ascending distance from the server.

In the first test, the jitter was negligible with less than 1% of packets deviating by more than 1ms from the mean latency. We see that as the geographical distance increases, the latency is slowly increasing as well but we also have to take into account network traffic which is not shown in the chart. As we move near the centre of London, the stations were busier with more people on the platform using their mobile devices. Until Chalk Farm, the latency and throughput are fairly constant but the big increase to the latency was found at London Bridge. Consequently, the throughput is also reduced. During the whole phase of the experiment, the traceroute showed a constant hop count of 12 hops with only the first hop (access point) changing at each station. This experiment did not provide conclusive results but it can be argued that as we move towards the centre of the city, networks are busier and the QoS drops. The nearly 2ms increase in latency between Chalk Farm and London Bridge cannot be considered a product solely of geographical distance since the distance between Chalk Farm and the server is approximately 5.1 miles while the distance between the server and London Bridge is 9.5 miles. If distance was the primary factor, the latency would have almost doubled. So in fact, we see that within metropolitan networks, the performance is not defined only by the distance but also by the load on the network. To better understand the structure of Virgin's network, a separate test was performed using a Virgin Home Broadband hotspot in Tufnell Park which is 5 miles away from the campus. In this test, the hop count was the same (albeit with different routers in the first 4 hops), as in the Tube's Wi-Fi but the latency was 17ms and the throughput was only 0.3MB/s. Traceroute results show that the higher latency for Virgin Home Broadband was caused by the first 4 hops that formed a different routing path to the one used by the Virgin's Wi-Fi in the Tube. We conclude that although the network is the same in both cases, the point of attachment is different and hence the subnets within the network that serve the client in each case, can exhibit different performance characteristics. This performance distinction between different points of attachment within the same network, demonstrates that there are determining factors that outweigh the geographical distance.

6.1.1.2 Manchester Wi-Fi Latency

To confirm the findings of the first experiment, a second series of tests was conducted from Manchester in order to study the effects of longer distances. The same methodology was used for the experiments but this time emphasis was given to latency and jitter. This time public Wi-Fi hotspots on BT's backhaul were used to connect to the Internet. The findings are presented in Fig. 6.2. The traceroute from each location is almost identical, and differentiated only by three routers which form the transit connection between JANET and BT's network. In all cases the total hop count was 17.



	Loc.1	Loc.2	Loc.3	Loc.4
🟦 Mean (ms)	51	546	46	399
🟧 Jitter (ms)	14	278	6	174
◼ Hops	17	17	17	17

Figure 6.2 Public Wi-Fi latency results in Manchester.

Once again, the results of the experiments do not show a linear relationship between latency and geographical distance. Manchester city centre is approximately 172 miles away from the server but under normal network conditions, the mean latency only increased 5 times. There are two locations in Manchester where the Wi-Fi network performed poorly and according to traceroute results, it was the metropolitan network at Manchester that caused the increased latency rather than Wi-Fi hotspot or BT's backbone that connects to JANET.

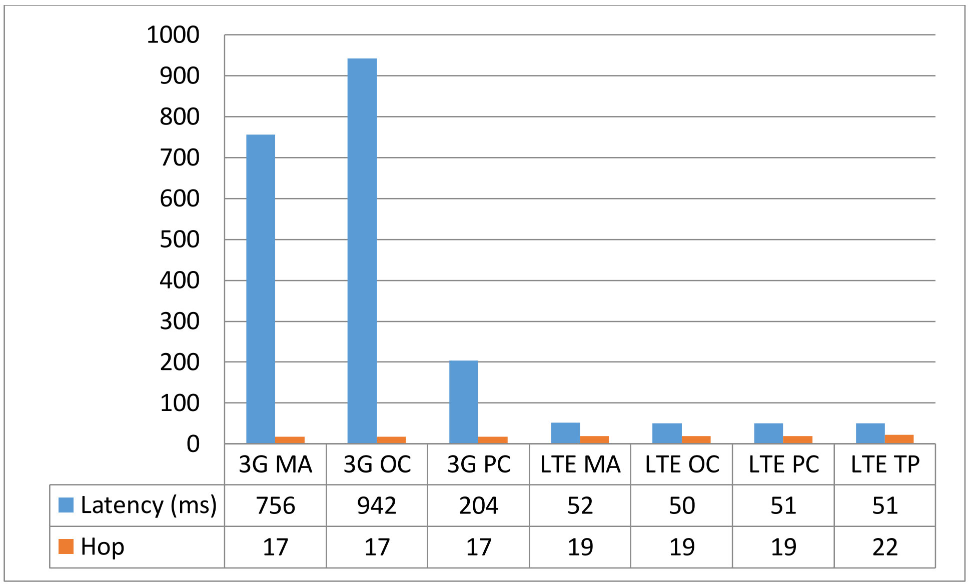
To clarify the causes of the increased hop count and latency between London and Manchester, another test was carried out using BT's network in order to understand the structure and the factors that are affecting performance. The test was conducted at Marble Arch in London and a latency of 19ms with a hop count of 15 was recorded. Between this test and the ones from Manchester, the only difference was the first 8 hops. While the connection from Marble Arch passed through London's Metropolitan network and reached the core of BT's network within 6 hops, the same connection from Manchester had to go through 8 hops before reaching BT's core. So although there is a small hop count increase to cover the extra distance, the excess latency was found to be caused by the metropolitan network in Manchester and more specifically, was caused by the first few hops of the connection. It is also interesting to note that the latency from Marble Arch using BT's network is higher than the latency from Virgin's network at London Bridge Tube Station despite Marble Arch being 3 miles closer to the server. Further tests performed by Middlesex University students using BT broadband also show that different points of attachment to the network exhibit different performance. The results are available for review but they are not included in this thesis.

6.1.1.3 International Wi-Fi Latencies

To further understand the impact of distance and different networks on the latency, the same test was conducted at two locations in Greece. The first location was in Piraeus, using a home broadband package from a local ISP and returned a latency of 90ms and a hop count of 14. This result is of particular interest since it returned latency higher than from Manchester but also returned a hop count smaller than the connection from Marble Arch. The second location was at the island of Aegina using a home broadband connection from the same ISP as in Piraeus. This test returned a hop count of 11 (which is even smaller than the free Wi-Fi in London's tube) and a latency of 99ms. Although the ISP was the same on both locations, the route to the server was vastly different with the route from the island using a different transit connection in the intercontinental backbone to reach JANET. Another interesting finding was that the route from Piraeus experienced path inflation by going through London, reaching Manchester and then going back to London to reach the server.

6.1.1.4 Mobile Network Latencies

After concluding the fixed broadband tests, a second series of tests was conducted using LTE and 3G in London in order to find out how mobile networks were structured and if there is any relationship between distance, hop count and latency. Due to data allowance constraints on mobile networks, it was impossible to use the same methodology as in the previous tests. Instead, the packet size remained at 1024 Bytes but only 1,000 tests were performed for a total of 1MB per test.



	3G MA	3G OC	3G PC	LTE MA	LTE OC	LTE PC	LTE TP
🟦 Latency (ms)	756	942	204	52	50	51	51
🟧 Hop	17	17	17	19	19	19	22

Figure 6.3 Mobile network latency results in multiple locations.

The first series of test was conducted using Vodafone's 3G network by converting a Smart Phone to a Wi-Fi hotspot. Measurements were taken at Marble Arch, Oxford Circus and Piccadilly Circus. The hop count was constant in all locations as shown in Fig. 6.3. Although Marble Arch is geographically closest to the server, the lowest recorded latency was at Piccadilly Circus. More interestingly, the first 4 hops of the routing path at Piccadilly Circus was different to that of the other locations. However, due to very high jitter and incomplete traceroute results due to some routers not responding to pings, it is not possible to determine how and why the performance improved so drastically at Piccadilly.

The second series of tests was conducting using EE's LTE network, once again by converting a Smart Phone to a Wi-Fi hotspot. The LTE network displayed much less jitter and more consistent performance compared to 3G. The hop count was also constant across the three locations in central London, and the latencies recorded in each location only differ by 2ms. The routing path was also consistent in those three locations. An extra test was conducted using LTE, at Tufnell Park for comparison with Virgin's Home Broadband from the same location. Once again, the LTE network recorded latency consistent with the locations in central London; however, the routing path changed and increased to 22 hops as shown in Fig. 5.3. Compared to Virgin Broadband results from Tufnell Park, the LTE connection has 10 more hops (22 vs 12) in the routing path to the server and three times the latency (51ms vs 17ms).

6.1.2 Summary of Latency Tests

What the above results show is that the Internet is a fluid landscape and the performance of networks cannot be easily determined by making associations with geographical distances and hops. In fact, even when using one network, different points of attachment to that network can offer greatly varying performance with only minor changes in the routing path. Therefore, what is needed to satisfy QoS demands for MCC is a mechanism for taking real-time measurements from networks and a model which we can use to determine if the offered QoS can satisfy the client's needs. Furthermore, since the main characteristic of 5G networks is constant connectivity via heterogeneous handovers, mobility has to be taken into account when modelling the performance of multiple networks along a user's path. The following section presents such a model.

6.2 QoS Evaluation Model for Mobile Clients

To evaluate the QoS of a single connection, a simple Markov chain can be used where the service request rate is defined as the number of service request sent by the client to the server such that satisfies the needs for the application without performance degradation. Therefore, the service request rate can be defined as the desirable flow of data in order to provide the desired level of performance from the application. The service rate is defined as the perceived rate of service responses that the client is receiving. Hence, the service rate is a factor of the Cloud's performance and the network's performance. Ideally, we want the service rate to be high enough so that it satisfies the request rate from the client. When this condition is not satisfied, requests are being queued by the network and consequently, the response time increases and application performance degrades. Fig. 6.4 visualises the Markov chain with a maximum buffer depth of 3 requests. The buffer depth can be scaled to any desired number to reflect different scenarios. We represent the service rate as μ and the request rate as λ.

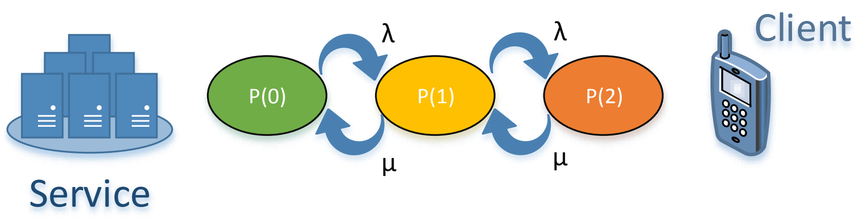


Figure 6.4 Markov Chain.

This model is currently employed as a method for evaluating the performance of networks and services at different service and request rates and buffer depths. However, in a mobility scenario, this model cannot be used to evaluate the performance of a connection because the network may change at any time as the user moves and thus λ may remain the same but μ can vary.

To describe a mobility scenario, we can consider a new queueing model, where multiple chains represent the different networks along the user's path. As the client moves and the device switches between networks, each chain represents the performance that the client experiences at their location. The client's movement can be expressed as a probability of "hopping" from one chain to another. This new model is illustrated in Fig. 6.5 in a 3x3 configuration. Each chain has its own service rate μ_n and also different probabilities for the client to leave the chain by moving to a chain upwards or downwards. We represent these probabilities as Mu_n and Md_n where n is always the originating chain. The general form of the model can be described as (N x M), where N defines the number of networks and M defines the buffer depth.

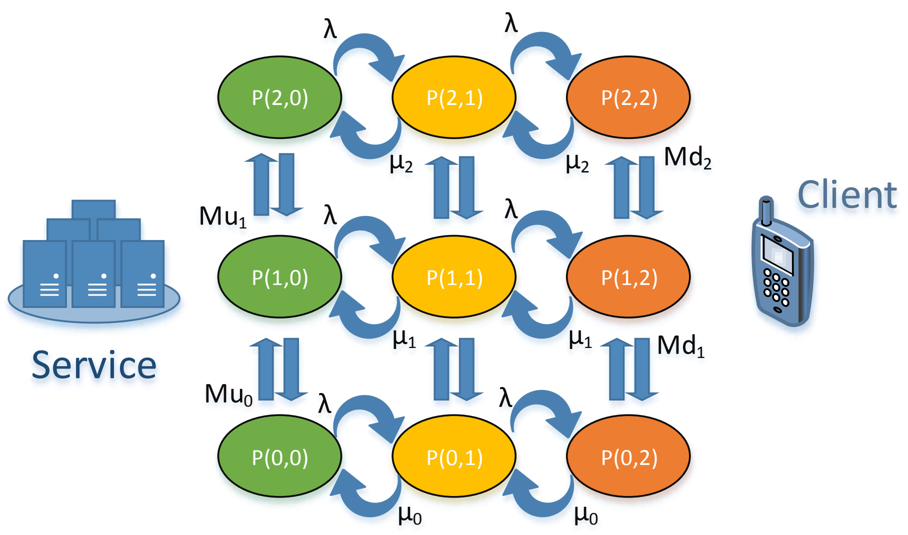


Figure 6.5 Multichain model in 3x3 configuration.

The probabilities of moving between chains can be derived by overlaying a map of wireless networks on the user's path as described in the previous chapter for the purposes of NDT. Different settings of the queueing model can be constructed for the user's path, with each one representing a different set of probabilities and combination of networks to which the mobile node may connect. Networks that demonstrate a high amount of queued requests present a potential problem for the QoE and in these cases, we can attempt to localise the MCC service to a datacentre within those networks so that performance may be improved by reducing the routing distance.

The different combinations of network and movement probabilities along a client's path can be represented as a multidimensional model, as shown in Fig. 6.6, with multiple planes, each one representing a different scenario and all of them intersecting at the location of the server. Alternatively, another way to use the model is to consider a set of chains as representing different points of attachment to the same network. This offers a more fine-grained approach as it gives us the ability to evaluate the performance of a single network from different points of attachment. For a real world example, we could consider this model applicable to an LTE network where each chain represents different base stations and potentially different subnets of the network. A vertical handover would be represented as a different set of chains and as long as the user remains within an administrative domain, the different points of attachment are represented as chains within the same set.

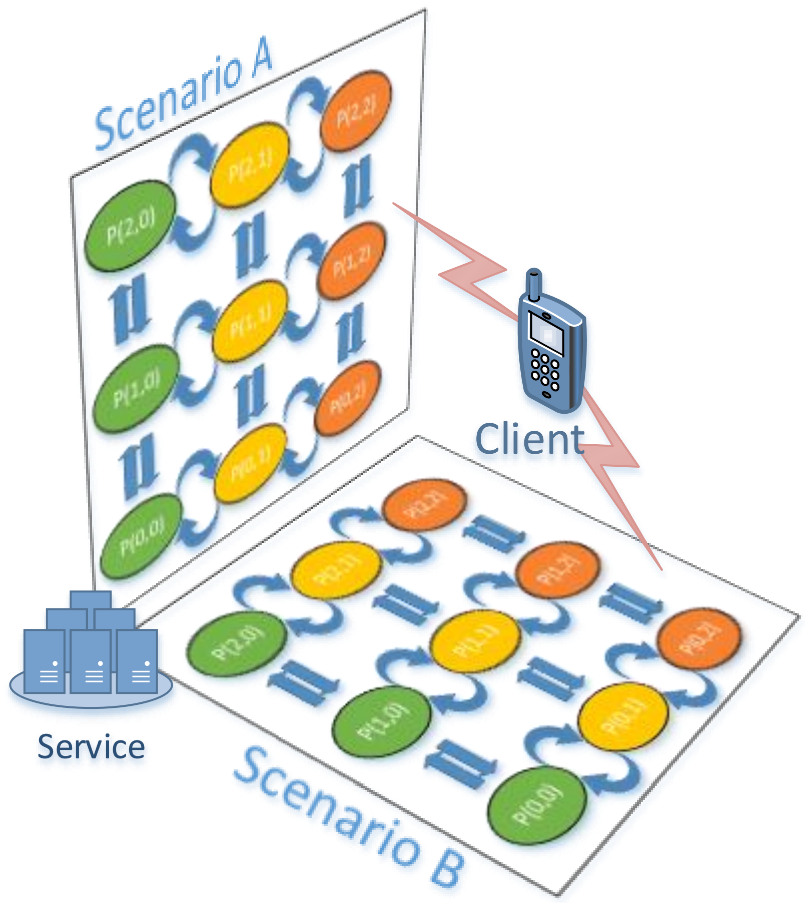


Figure 6.6 Multi-dimensional model based on a 3x3 chain configuration.

This model can be used in two different modes of operation in the context of MCC and 5G networks. The first mode is to use the model for making network selection decisions. This way we can evaluate the performance of networks along a user's path and create a list of preferred networks that the mobile node will prioritise in the selection process. The second mode of operation is to use this model as a mechanism for dynamically localising services when a network is insufficient performance is found on the user's path and presents the only available connectivity option. This approach heavily depends on measuring the latency of networks as well as identifying where most of the latency is caused. If caused by the user's access point or another router within the user's network, localising the service would marginally improve the QoE. This mode mostly applies to scenarios where the backbone (i.e. transit and peer connections) is the cause of latency and hence, localising a service would bypass the problematic networks and improve the performance. However, we can also consider an extreme case where the latency does originate from the user's network but the performance is so low that as an emergency action, the service is localised in order to reduce as much latency as possible from other networks.

6.2.1 Operational Requirements and Limitations

In either mode of operation, in order for this model to function, a set of requirements has to be satisfied. The first requirement is that the application of the user can define the required service rate. This is not necessarily in terms of bandwidth of latency and it can be in terms of frame-rate so that different applications can define their desired level of responsiveness. In this case, the estimated size of each frame in terms of data will have to be reported as well so that the network can determine if the latency and throughput are adequate.

The second requirement is that network QoS can be reported in advance (i.e. before the client connects to each network) so that the model can evaluate the total performance of different networks along the user's path. This requires a set of mechanisms that probe networks and gather performance information for the current location of the user's service.

The third requirement is that the networks can report the user's path so that probabilities of handovers can be calculated in advance along with the user's NDT for each network.

This model does not take into account any performance degradation due to handovers. For instance, any additional latency induced by the process of the handover, is not considered as a factor in the performance of the networks. Therefore, the results presented here may be different to those from a real world scenario.

Due to the large number of variables introduced to the model when it is being scaled to NxM, it is very difficult to derive a closed-form solution. This means that we cannot take this model and apply it as a general solution to multiple networks. Each network, will need to solve the model for the amount of NxM that they desire, depending on how far into the future they want to predict a user's path and how big a buffer they desire. After the model is solved for any NxM values, the user's mobility rate and performance inputs can be applied to the equations to model the performance of a connection.

6.3 Solving the Model

The queueing model can be solved mathematically for any (NxM) configuration, however, as the size of the model increases, the number variables also increases, making it very hard to derive a closed-form solution. In this section, we will look at the methodology used for solving the queueing model in a 2x3 configuration as presented in Fig. 6.7.

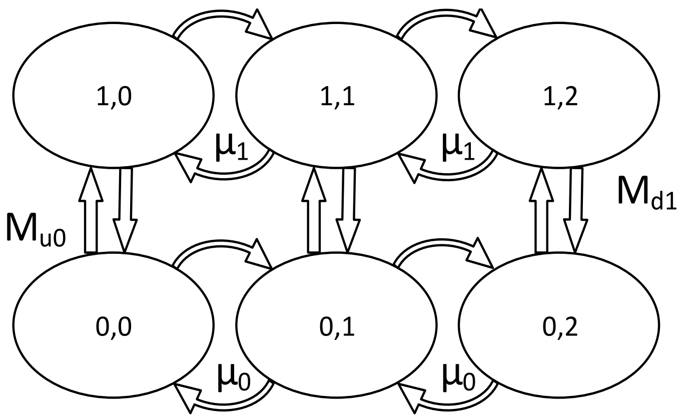


Figure 6.7 2x3 queueing model.

The balance equations:

We start by constructing the balance equations. In this process, we express a balance between anything going in and anything going out of each state of the mode. Thus for each state, we have:

$$(M_{u0} + \lambda)P_{0,0} = \mu_0 P_{0,1} + M_{d1} P_{1,0} \tag{6.1}$$

$$(M_{u0} + \lambda + \mu_0)P_{0,1} = M_{d1} P_{1,1} + \mu_0 P_{0,2} + \lambda P_{0,0} \tag{6.2}$$

$$(M_{u0} + \mu_0)P_{0,2} = M_{d1} P_{1,2} + \lambda P_{0,1} \tag{6.3}$$

$$(M_{d1} + \lambda)P_{1,0} = M_{u0} P_{0,0} + \mu_1 P_{1,1} \tag{6.4}$$

$$(M_{d1} + \lambda + \mu_1)P_{1,1} = M_{u0}P_{0,1} + \mu_1 P_{1,2} + \lambda P_{1,0} \tag{6.5}$$

$$(M_{d1} + \mu_1)P_{1,2} = M_{u0}P_{0,2} + \lambda P_{1,1} \tag{6.6}$$

We then proceed to unzip the model by expressing each state as a function of its adjacent states. Starting from **P$_{2,2}$**, we have:

$P_{1,2} = a_{1,2}P_{0,2} + b_{1,2}P_{1,1}$

$\frac{M_{u0}}{(M_{d1}+\mu_1)} = a_{1,2}$ $\quad \frac{\lambda}{(M_{d1}+\mu_1)} = b_{1,2}$

Now we can express **P$_{1,1}$** by substituting **P$_{1,2}$**:

$\boldsymbol{P_{1,1}} = a_{1,1}P_{0,1} + b_{1,1}P_{0,2} + c_{1,1}P_{1,0}$

$\frac{M_{u0}}{(M_{d1}+\lambda+\mu_1-\mu_1 b_{1,2})} = a_{1,1}$ $\quad \frac{\mu_1 a_{1,2}}{(M_{d1}+\lambda+\mu_1-\mu_1 b_{1,2})} = b_{1,1}$ $\quad \frac{\lambda}{(M_{d1}+\lambda+\mu_1-\mu_1 b_{1,2})} = c_{1,1}$

We rewrite **P$_{1,2}$** by substituting the value for **P$_{1,1}$**:

$P_{1,2} = (a_{1,2} + b_{1,2}b_{1,1})P_{0,2} + b_{1,2}a_{1,1}P_{0,1} + b_{1,2}c_{1,1}P_{1,0}$

Same process for **P$_{1,0}$**:

$\boldsymbol{P_{1,0}} = a_{1,0}P_{0,0} + b_{1,0}P_{0,1} + c_{1,0}P_{0,2}$

$\frac{M_{u0}}{(M_{d1}+\lambda-\mu_1 c_{1,1})} = a_{1,0}$ $\quad \frac{\mu_1 a_{1,1}}{(M_{d1}+\lambda-\mu_1 c_{1,1})} = b_{1,0}$ $\quad \frac{\mu_1 b_{1,1}}{(M_{d1}+\lambda-\mu_1 c_{1,1})} = c_{1,0}$

We rewrite **P$_{1,1}$** and **P$_{1,2}$** once more using the new substitute equations:

$\boldsymbol{P_{1,1}} = (a_{1,1} + c_{1,1}b_{1,0})P_{0,1} + (b_{1,1} + c_{1,1}c_{1,0})P_{0,2} + c_{1,1}a_{1,0}P_{0,0}$

$\boldsymbol{P_{1,2}} = (a_{1,2} + b_{1,2}b_{1,1} + b_{1,2}c_{1,1}c_{1,0})P_{0,2} + (b_{1,2}a_{1,1} + b_{1,2}c_{1,1}b_{1,0})P_{0,1} + b_{1,2}c_{1,1}a_{1,0}P_{0,0}$

We now move to the lower chain, starting from **P$_{0,2}$**, we create the following expression by substituting **P$_{1,2}$**:

$P_{0,2} = a_{0,2}P_{0,1} + b_{0,2}P_{0,0}$

$\frac{(M_{d1}b_{1,2}a_{1,1}+M_{d1}b_{1,2}c_{1,1}b_{1,0}+\lambda)}{(M_{u0}+\mu_0-M_{d1}a_{1,2}-M_{d1}b_{1,2}b_{1,1}-M_{d1}b_{1,2}c_{1,1}c_{1,0})} = a_{0,2}$ $\quad \frac{M_{d1}b_{1,2}c_{1,1}a_{1,0}}{(M_{u0}+\mu_0-M_{d1}a_{1,2}-M_{d1}b_{1,2}b_{1,1}-M_{d1}b_{1,2}c_{1,1}c_{1,0})} = b_{0,2}$

We can now go back and substitute $\mathbf{P_{0,2}}$ in the upper chain:

$$\boldsymbol{P}_{\mathbf{1,0}} = (a_{1,0} + c_{1,0}b_{0,2})P_{0,0} + (b_{1,0} + c_{1,0}a_{0,2})P_{0,1}$$

$$\boldsymbol{P}_{\mathbf{1,1}} = (a_{1,1} + c_{1,1}b_{1,0} + a_{0,2}b_{1,1} + a_{0,2}c_{1,1}c_{1,0})P_{0,1} + (b_{1,1}b_{0,2} + b_{0,2}c_{1,1}c_{1,0} + c_{1,1}a_{1,0})P_{0,0}$$

$$\boldsymbol{P}_{\mathbf{1,2}} = (a_{1,2}a_{0,2} + b_{1,2}b_{1,1}a_{0,2} + a_{0,2}b_{1,2}c_{1,1}c_{1,0} + b_{1,2}a_{1,1} + b_{1,2}c_{1,1}b_{1,0})P_{0,1} +$$
$$(a_{1,2}b_{0,2} + b_{1,2}b_{1,1}b_{0,2} + b_{1,2}c_{1,1}c_{1,0}b_{0,2} + b_{1,2}c_{1,1}a_{1,0})$$

We can now express $\mathbf{P_{0,1}}$ as follows:

$$\boldsymbol{P}_{\mathbf{0,1}} = uP_{0,0} \tag{6.7}$$

$$\frac{(M_{d1}b_{1,1}b_{0,2}+M_{d1}b_{0,2}c_{1,1}c_{1,0}+M_{d1}c_{1,1}a_{1,0}+\mu_0 b_{0,2}+\lambda)}{(M_{u0}+\lambda+\mu_0-M_{d1}a_{1,1}-M_{d1}c_{1,1}b_{1,0}-M_{d1}a_{0,2}b_{1,1}-M_{d1}a_{0,2}c_{1,1}c_{1,0}-\mu_0 a_{0,2})} = u$$

And thus, we can write the final form for each state as a function of $\mathbf{P_{0,0}}$:

$$\begin{aligned}\boldsymbol{P}_{\mathbf{1,2}} = (&ua_{1,2}a_{0,2} + ub_{1,2}b_{1,1}a_{0,2} + ua_{0,2}b_{1,2}c_{1,1}c_{1,0} + ub_{1,2}a_{1,1} + ub_{1,2}c_{1,1}b_{1,0} \\ &+ a_{1,2}b_{0,2} + b_{1,2}b_{1,1}b_{0,2} + b_{1,2}c_{1,1}c_{1,0}b_{0,2} + b_{1,2}c_{1,1}a_{1,0})P_{0,0}\end{aligned} \tag{6.8}$$

$$\begin{aligned}\boldsymbol{P}_{\mathbf{1,1}} = (&ua_{1,1} + uc_{1,1}b_{1,0} + ua_{0,2}b_{1,1} + ua_{0,2}c_{1,1}c_{1,0} \\ &+ b_{1,1}b_{0,2} + b_{0,2}c_{1,1}c_{1,0} + c_{1,1}a_{1,0})P_{0,0}\end{aligned} \tag{6.9}$$

$$\boldsymbol{P}_{\mathbf{1,0}} = (a_{1,0} + c_{1,0}b_{0,2} + b_{1,0}u + uc_{1,0}a_{0,2})P_{0,0} \tag{6.10}$$

$$\boldsymbol{P}_{\mathbf{0,2}} = (a_{0,2}u + b_{0,2})P_{0,0} \tag{6.11}$$

Finally, the sum of all probabilities is equal to 1 and therefore we have:

$$\mathbf{1} = \boldsymbol{P}_{\mathbf{0,0}} + \boldsymbol{P}_{\mathbf{0,1}} + \boldsymbol{P}_{\mathbf{0,2}} + \boldsymbol{P}_{\mathbf{1,0}} + \boldsymbol{P}_{\mathbf{1,1}} + \boldsymbol{P}_{\mathbf{1,2}} \tag{6.12}$$

$$\mathbf{1} = \boldsymbol{P}_{\mathbf{0,0}} + u\boldsymbol{P}_{\mathbf{0,0}} + (a_{0,2}u + b_{0,2})\boldsymbol{P}_{\mathbf{0,0}} + (a_{1,0} + c_{1,0}b_{0,2} + b_{1,0}u + uc_{1,0}a_{0,2})\boldsymbol{P}_{\mathbf{0,0}} + (ua_{1,1} + \mathrm{u}c_{1,1}b_{1,0} + ua_{0,2}b_{1,1} + ua_{0,2}c_{1,1}c_{1,0} + b_{1,1}b_{0,2} + b_{0,2}c_{1,1}c_{1,0} + c_{1,1}a_{1,0})\boldsymbol{P}_{\mathbf{0,0}} + (ua_{1,2}a_{0,2} + ub_{1,2}b_{1,1}a_{0,2} + ua_{0,2}b_{1,2}c_{1,1}c_{1,0} + ub_{1,2}a_{1,1} + \mathrm{u}b_{1,2}c_{1,1}b_{1,0} + a_{1,2}b_{0,2} + b_{1,2}b_{1,1}b_{0,2} + b_{1,2}c_{1,1}c_{1,0}b_{0,2} + b_{1,2}c_{1,1}a_{1,0})\boldsymbol{P}_{\mathbf{0,0}}$$

$$\boldsymbol{P}_{\mathbf{0,0}} = \mathbf{1}/(1 + u + a_{0,2}u + b_{0,2} + a_{1,0} + c_{1,0}b_{0,2} + b_{1,0}u + uc_{1,0}a_{0,2} + ua_{1,1} + \mathrm{u}c_{1,1}b_{1,0} + ua_{0,2}b_{1,1} + ua_{0,2}c_{1,1}c_{1,0} + b_{1,1}b_{0,2} + b_{0,2}c_{1,1}c_{1,0} + c_{1,1}a_{1,0} + ua_{1,2}a_{0,2} + ub_{1,2}b_{1,1}a_{0,2} + ua_{0,2}b_{1,2}c_{1,1}c_{1,0} + ub_{1,2}a_{1,1} + ub_{1,2}c_{1,1}b_{1,0} + a_{1,2}b_{0,2} + b_{1,2}b_{1,1}b_{0,2} + b_{1,2}c_{1,1}c_{1,0}b_{0,2} + b_{1,2}c_{1,1}a_{1,0})$$

$\mathbf{P_{0,0}}$ is now expressed as a function of all the variables we can input. Every other state probability is calculated based on $\mathbf{P_{0,0}}$. The sum of all probabilities for any input values will always be equal to 1.

6.4 Test Scenarios and Assumptions

To help understand how the model may be applied, this section presents two different mobility scenarios that can be studied using the 3x3 model. The two scenarios analyse Random and Fixed-Path mobility which are the two prominent types of mobility. Before analysing the scenarios, we will look at some of the prominent methods for determining user mobility which is one of the main inputs for the model.

6.4.1 Measuring Mobility

For the purposes of traffic management, we introduced NDT as a method of calculating the time that a user will stay within a network and at the same time, as an indication of the probability of joining other networks under the right conditions. Because the queueing model takes the rate of mobility as input, NDT does not apply directly. In this section, we introduce a method for measuring the rate of mobility and a proposed solution for determining the sequence of networks along a user's path.

A cell's outgoing rate can be expressed as a function of the radius, the circumference and the area of the cell and the average velocity of the user (Liu et al. 2006) [41]. This is expressed as follows:

$$M_{cdwell} = \frac{E(v)L}{\pi A}$$

(6.13)

Where, $E(v)$ is the average velocity of the user, L is the length of the cell's perimeter and A is the area of the cell. We can substitute L and A with $2\pi R$ and πR^2 respectively, where R is the radius of the circle. We can now rewrite the expression as follows:

$$M_{cdwell} = \frac{2E(v)}{\pi R}$$

(6.14)

In addition to measuring the mobility rate, we also need to identify which network the user will move to next. As explained before, we can predict this based on NDT calculations and an overlay map of networks. An alternative method, which would more accurately predict upcoming network connections, is described by Mapp et al. (2012) [44]. They investigate the use of a service that monitors a user's mobile device connectivity and gathers information about each network that the device joins. The information is stored on a server along with contextual information about user mobility, such as which network the device was connected to previously and to which network it handed over afterwards. This Wireless Footprint method can provide mobility information based on a user's past mobility patterns. Therefore, we can use this method for estimating the next network that the user may join and thus create a proactive system for determining the QoS. The following subsections present three different mobility scenarios as representative examples of different types of mobility which help demonstrate the functionality of the proposed model. Real life human mobility would be characterised as a combination of these scenarios, however the exact characteristics would be the outcome of a study dedicated on the subject and therefore are outside the context of this chapter.

6.4.2 A Case of Urban and Fixed-Path Mobility

In the case of urban and fixed-path mobility, we can use this simple 2x3 model to evaluate the performance of networks as the user passes through them. We will assume that Wireless Footprinting is not used and therefore, the 2x3 model is preferable in this case as it is easier to predict the next location of the user based on their position and movement. A buffer depth of 3 was chosen for this particular case as it easy to solve and can give a good example of how the model works.

Urban Mobility (Random Movement)

We start with an urban mobility example, where the user has equal probability of moving back or forward and therefore the device may be switching between two networks at an equal rate. Another way of viewing this example, is to consider that the user may not be moving back and forth, however, in his direction of movement, there are alternating network areas of two different providers. Therefore, from the network's perspective, the user exhibits a palindromic movement between two networks, or in other words, the user may have a fixed direction but the device hops between two networks in an alternating manner. The inputs to the model are presented in Table 6.1. We choose unequal service rate between networks to visualise the impact to the overall performance when the device handovers between networks of different QoS. The mobility rates were derived from equation (6.14) using a radius of 50 metres for the network and a human walking speed of 5km/h.

Table 6.1 Random urban mobility inputs with one network unable to provide the required QoS.

λ	μ0	μ1	Mu0	Md1
60.0000	30.0000	60.0000	0.0177	0.0177

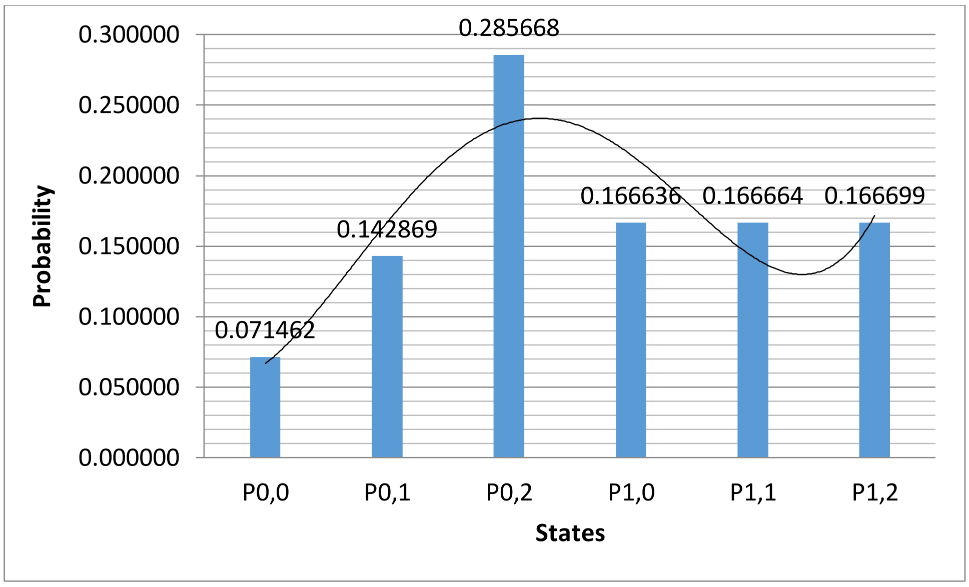


Figure 6.8 Random urban mobility results.

In this scenario, we see that the first network (**μ0**) has half the service rate of the second network (**μ1**). What is interesting in this case, is that the model tells us that the low performance of the first network has a detrimental effect on the performance of the second network. Under normal conditions, the second network would have equal probabilities across all its states ($P_{1,M}$), however, we see that there is a slightly elevated probability of being in the queue. This is a result of queued requests from the first network, coming through to the second network. When the mobility is random and the user is switching between these two networks at an equal rate, the performance of the first network improves slightly due to requests being served by the second network, but the performance of the second network degrades slightly due to the extra requests that were pending on the first network. We now look at the same scenario but with networks of adequate performance as shown in Table 6.2.

Table 6.2 Random urban mobility with sufficient QoS on both networks.

λ	μ0	μ1	Mu0	Md1
60.0000	65.0000	80.0000	0.0176	0.0176

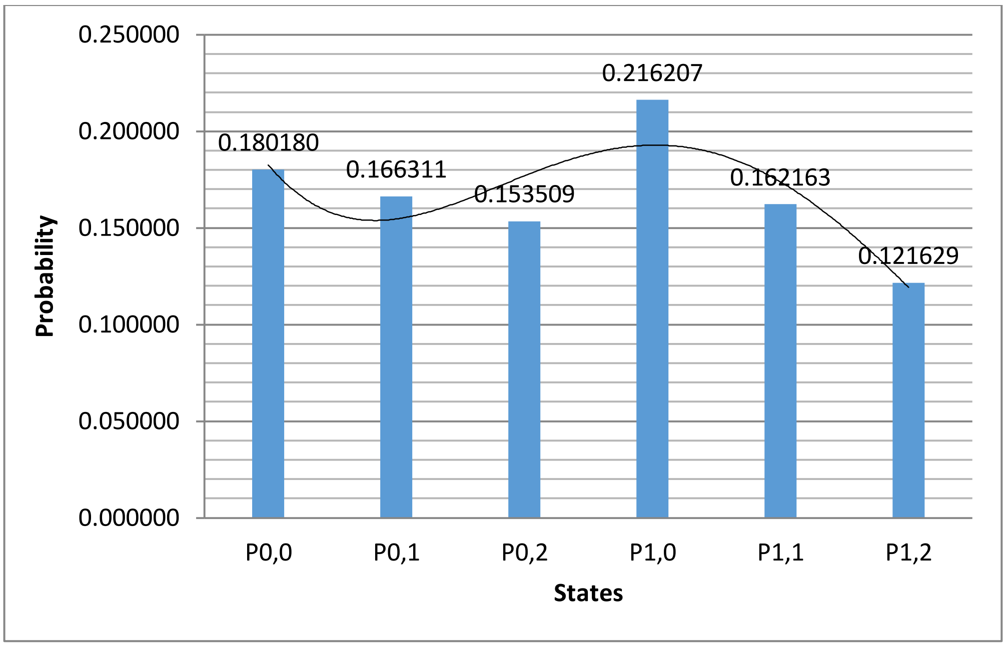


Figure 6.9 Random urban mobility results. Both networks have adequate QoS.

We see from Fig. 6.9 that both networks have adequate performance meaning that there is a higher probability of having an empty queue ($P_{0,0}$ and $P_{1,0}$) as opposed to having queued requests. In this case, the model tells us that both networks can successfully provide the requested services at very high performance levels.

Urban Mobility (Fixed Path)

Next, we look at a fixed-path mobility example where the user is moving in one direction and therefore there is a very small chance of returning to a previous network. The inputs for this scenario are presented in Table 6.3. Once again, we start with a case where one of the networks has insufficient QoS. The values to derive the upwards mobility rate were set to 80km/h for velocity and 500 metres cell radius

Table 6.3 Fixed-path mobility. The first network has insufficient QoS.

λ	μ0	μ1	Mu0	Md1
60.0000	30.0000	60.0000	0.0283	0.0001

In this scenario, we would expect the second network to have equal probabilities across all the states since **μ1** equals **λ**. However, as shown in Fig. 6.10, the performance of the second network is once again slightly affected by the first network. We find the probability of being idle ($P_{1,0}$) to be lower than being in the queue. This does not happen when we equalise **μ0** and **μ1** while leaving all the other values as in Table 6.3**.** Fig. 6.11 illustrates this.

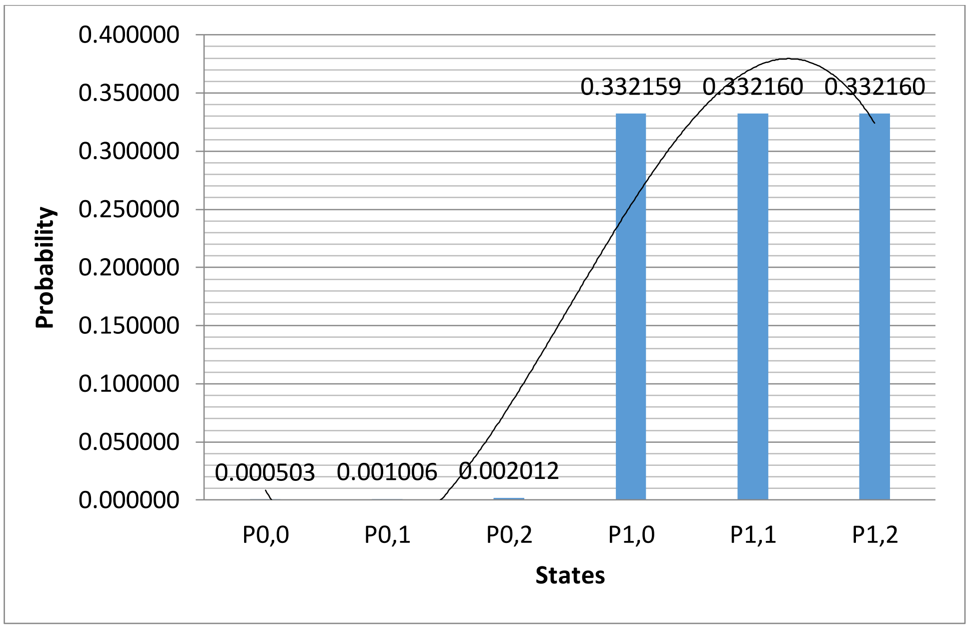


Figure 6.10 Fixed-path mobility with insufficient QoS on first network.

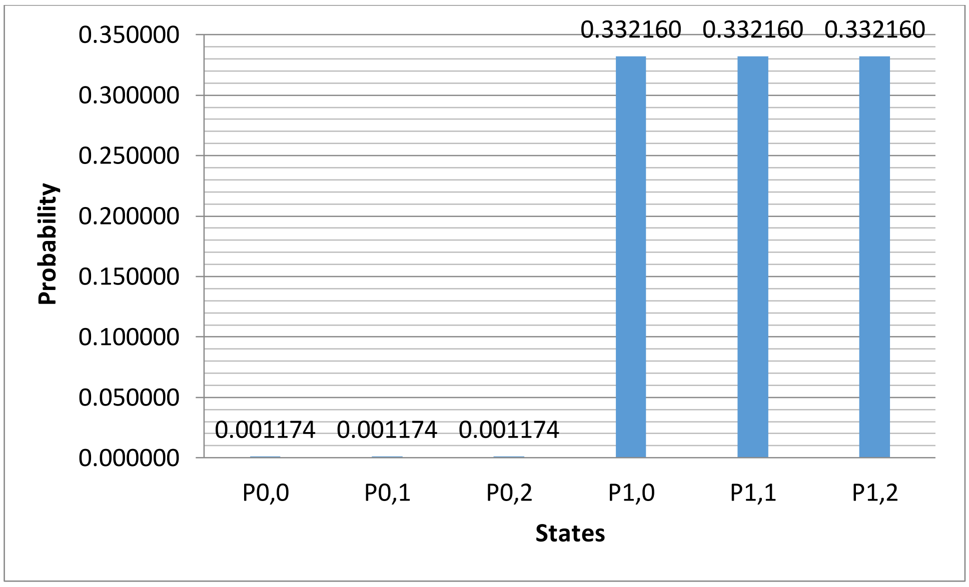


Figure 6.11 Fixed-path mobility with equal QoS on networks.

We see from these scenarios that mobility plays a role to the overall QoS even when ignoring performance degradation from the handover events. Therefore, when modelling the performance of MCC services, user mobility has to be taken into account, and where possible, network performance should be adjusted to compensate, either via network selection mechanisms or via service localisation. In the next section, we look at an example where Wireless Footprinting can provide information about a sequence of networks.

6.4.3 A Case of Contextual Mobility

Wireless Footprinting gives us the ability to determine the sequence of networks that to which a device will attach. Consequently, we can predict with a certain level of confidence, how service requests will be distributed across multiple networks along a user's path. Therefore, we can expand the 2x3 model by adding an extra chain to represent a network further down the user's path. So in this case, let us investigate the performance of the 3x3 model in a scenario of contextual mobility. The solution for the 3x3 model can be found in Appendix B. The input values are presented in Table 6.4 and the representation of the model is in Fig. 6.5. For this scenario, we consider a speed of 50km/h for the user and a cell radius of 1km, this representing motorway mobility.

Table 6.4 Contextual mobility inputs for fixed-path.

λ	μ0	μ1	μ2	Mu0	Md1	Mu1	Md2
60.0000	80.0000	40.0000	20.0000	0.0088	0.0001	0.0088	0.0001

As we see from the results in Fig. 6.12, the user is bound to end up on the third network which has insufficient service rate and consequently, the performance of their connection will degrade as the requests will be queued. At the same time, we also see from the results that the first network had better performance since the probability of being idle is higher than having any queued requests. Finally, the second network also shows higher probability of queueing requests and therefore the performance there is also insufficient.

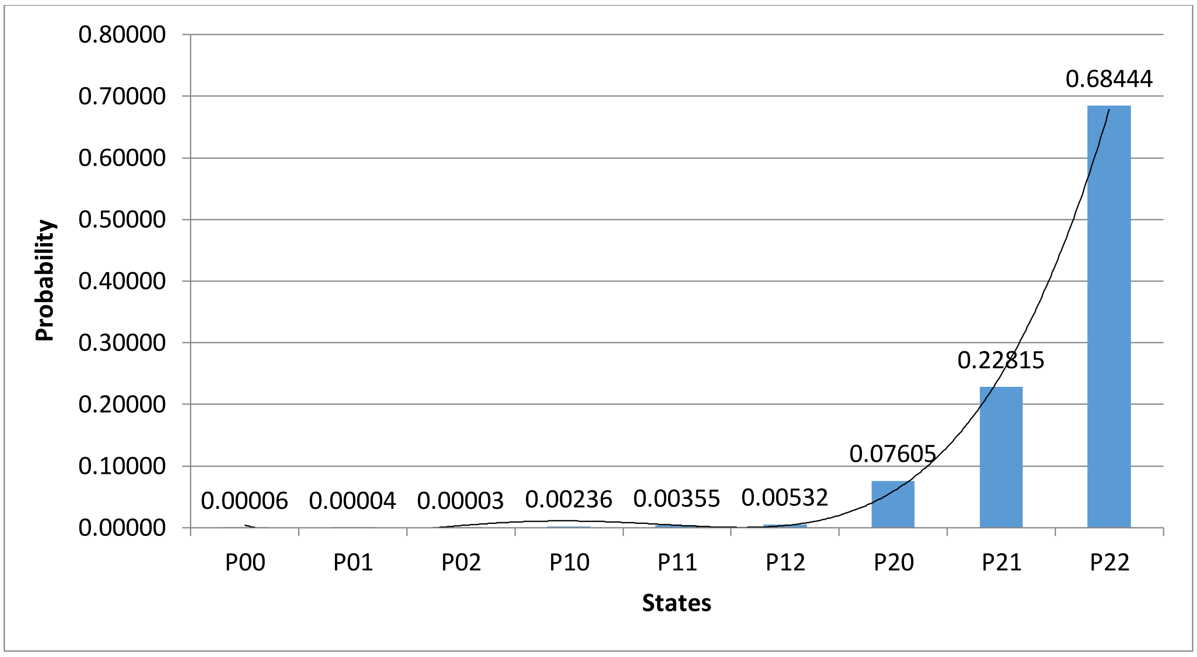


Figure 6.12 Contextual mobility results for fixed-path.

Another potential use case for the 3x3 model using Wireless Footprinting would involve a scenario where a user has equal probability of joining two different networks. We can envision such a scenario by considering a user currently connected to an LTE networks and moving towards an area where multiple overlapping Wi-Fi networks can be found. From past records, Wireless Footprinting may report that the device has equal probability of joining either network, so in this case, we want to use the model to select the network with adequate performance for the user's applications. So let us consider a person walking while their device is streaming music. We set the walking speed to 5km/h and the radius of the Wi-Fi and LTE network to 60 and 500 metres respectively. Table 6.5 contains the inputs for this scenario.

Table 6.5 Contextual mobility with two probable targets.

λ	μ0	μ1	μ2	Mu0	Md1	Mu1	Md2
40.0000	55.0000	45.0000	65.0000	0.0150	0.0018	0.0018	0.0150

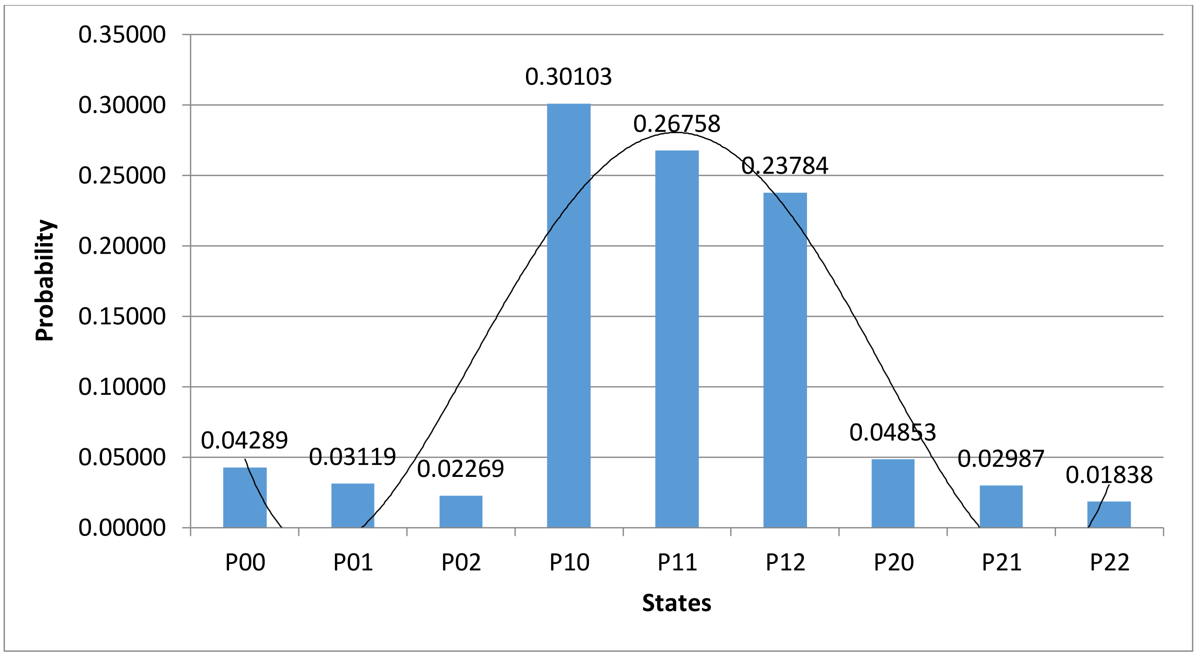


Figure 6.13 Contextual mobility results for multiple networks.

In this case, we consider the user to be starting from the middle chain and based on the probabilities given by Wireless Footprinting and the movement rate of the user, we discover that they may move either to the top or the bottom chain of the model. So in this case the movement rate is equal for going up or down in the chain but it is also very likely that the user will return to the LTE network upon leaving the range of the Wi-Fi networks. Fig. 6.13 shows the results from this scenario and we see that the performance of the top network is better than the performance of the other two. The LTE network is the least capable of maintaining a high QoS so the mobile device could choose to hop to either of the Wi-Fi networks. Since we want to achieve efficient utilisation of resources, there are two options: the first option is to move to the first network (bottom chain) and temporarily stream music from there. The second option is to move to the second network (top chain) and stream music from there while also trying buffer as much music as possible so that upon returning to the LTE network, some of the queueing delays may be absorbed by the cached music. This scenario demonstrates how this model can be used as a network selection mechanism.

6.4.4 Scenario-Based Application

In order to apply this model to a real world scenario, we need to construct multiple scenarios that represent different mobility estimations. For example, if we wish to apply the 2x3 model to a case where the device may have the option of performing a handover between multiple networks, we would need to construct a scenario for each network such that the bottom chain of the model represents the current network of the user and the top chain represents a potential target. We can then apply different probabilities via Wireless Footprinting to each scenario to find out where the handover is more probable. Fig. 6.14 demonstrates how various scenarios may be constructed. Additionally, we can use the model to analyse the overall performance of each scenario so that when multiple scenarios are equally likely, the network will automatically select the scenario with better performance or more efficient utilisation of resources. Thus, we can apply this model as a network selection mechanism.

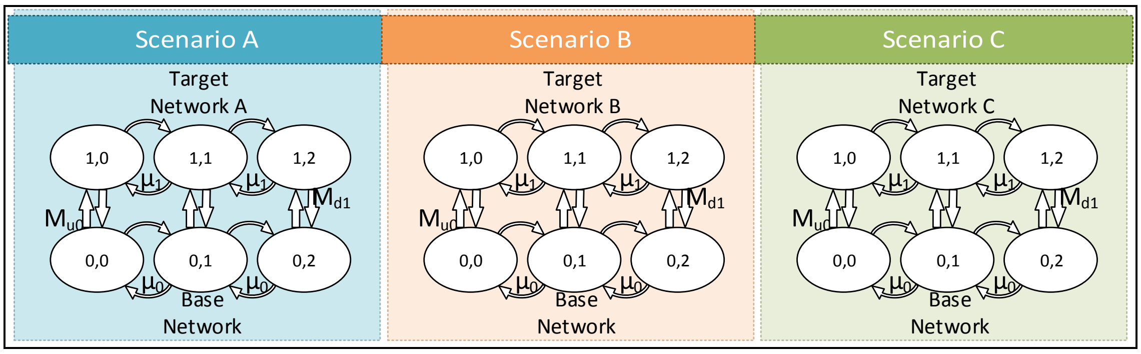


Figure 6.14 Illustration of multiple scenarios of handovers.

Alternatively, after performing the above calculations, if the highest probability applies to a scenario with suboptimal performance, we could calculate the user's NDT for the target network in that particular scenario and use that to determine if there is enough time to localise the service to that network. Ultimately this scenario is limited by how far into the future, the location of the user can be predicted but what is also important to consider is that localising a service to achieve good performance within one network, may actually have negative effects on the performance of other networks. This is particularly applicable if the network where the service was localised, does not have adequate peering resources to other networks.

6.5 Critical Summary

Looking back at the proposed service delivery framework, this mechanism falls in the service delivery layer along with the traffic management mechanism. Considering the above scenarios and the information presented about the structure of the Internet, we can potentially envision particular scenarios where traffic management and QoS provisioning may be in conflict. For example, the QoS management system may suggest a service localisation because it thinks that it is necessary and the user's NDT may be adequate to perform the migration, but the traffic management system may object the decision by claiming that the NDT may be adequate for localising a service but insufficient for creating any traffic savings, thus adding load to the network.

The proposed model does not appear to have a general closed-form solution because the different networks have different service rates and arrival rates as well as different mobility rates; therefore, this limits the applicability of the model to real-life scenarios. Practically, this limits the model to using a small number of networks simultaneously and hence an interactive approach would be needed for a more complete solution.

Finally, this model raises the issue of identifying a target datacentre for a user's mobility pattern. Mechanisms to identify target datacentres need to consider the performance that the datacentre can achieve as well as the performance of the network that connects the datacentre to the client. Furthermore, it would be beneficial if the selection mechanism would take into account the peering relationship between the datacentre and other networks that the user will encounter along their path. The datacentre selection mechanism is therefore more complex in the context of QoS than in the context of traffic localisation within an AS.

The relationship between the traffic and QoS management systems explored in this thesis has to be ultimately determined by the user and the network operators. Some users may wish to give priority to QoS, while users without preference may default to pre-assigned traffic management heuristics. These details can be defined in the Service Management and Service Subscription layers of the framework. In conclusion, the relative weight of each mechanism in the decision-making process has to be determined by the users, the network operators or the service providers and it will be driven by factors such as costs and resource utilisation.

Chapter 7 Conclusion and Future Work

In this chapter, a summary of the completed work is presented and the major contributions to knowledge are highlighted. This is followed by the proposed future work as well as work currently undertaken to create a traffic monitoring mechanism. The chapter concludes with a closing statement of the author's views on future Internet and Cloud technology.

7.1 Summary of Work

This thesis presents the evolution of thin-client computing by focusing on examples of task offloading in a mobile environment for the purpose of expanding and enhancing the built-in capabilities of mobile devices. Chapter 2 investigated this evolution from early attempts at offloading mobile tasks on localised surrogate nodes to modern Mobile Cloud Computing technology. It was highlighted that although Cloud technology can be a more powerful and efficient solution for supporting mobile devices, network performance issues that are not prominent in localised surrogate approaches became more important when offloading tasks to a remote Cloud.

Chapter 3 presented the evolution of the Internet and how modern technologies as well as the modern highly interconnected structure of the Internet are helping to deliver lower latencies and higher bandwidths. Focus was placed on mobile networks and problems that arise from user mobility as devices switch between different network providers and therefore experience different network performance. The chapter concluded that in order to better support modern mobile devices on the Internet, a framework that takes into account network performance and user mobility is necessary.

Chapter 4 proposed a new service delivery framework focused on improving service delivery for mobile clients using Cloud technology. This framework takes into account recent efforts in the area of seamless connectivity across heterogeneous networks, as well as capabilities of Cloud technology. In compliance with the definition of Cloud technology, the framework allows for on-demand resource allocation after a negotiation process between Cloud datacentres and also allows elastic and dynamic resource pooling based on application requirements and user demands.

Using this framework as a guide, the thesis explored the dynamic localisation of personalised Cloud services for the purpose of traffic management in a mobile environment. The experimental results from the prototype solution as presented in Chapter 5 show that it is possible from the network's perspective to create mechanisms for service localisation based on user mobility and network traffic throughput and such mechanisms can achieve considerable gains in the aspect of traffic management under heavy application usage scenarios such as Cloud gaming.

Chapter 6 investigated a model for estimating the overall performance of networks along a mobile user's path and uses it to determine when a service should be moved to a different network when QoS demands are not satisfied. The two proposed solutions in chapters 5 and 6 can be used in tandem as part of the proposed service delivery model in order to maximise QoS and localise traffic.

7.2 Contribution to Knowledge

This thesis presents and explores service localisation for mobile users as a means of improving the QoS and managing network traffic. The proposed Service Delivery framework takes into account user and service requirements and characteristics and combines them with network conditions and user mobility to determine the best way to deliver a service. The novelty of the presented framework lies in the convergence of Cloud technology, 5th generation networks and user mobility.

For the purposes of traffic management, a set of mathematical equations was developed as part of a mechanism that estimates the minimum NDT required to achieve traffic savings through localisation. The equations are implemented as part of a prototype system that dynamically moves a VM based on traffic throughput. The experimental prototype is a novel approach to service orchestration based on Economic Traffic Management rules and user mobility.

For the purposes of QoS provisioning, this thesis proposed a multi-dimensional queuing model that takes into account user mobility and models the overall QoS of multiple networks in the user's path. Existing queueing models take into account user mobility from the perspective of each network and consider mobility only to the extent it affects the performance of a single network. The proposed network considers the overall end-to-end performance across multiple networks depending on how user mobility affects the rate of network handovers and weighs the impact that each network has according to how long the device will remain connected to it. This model can be used either as a network selection mechanism for 5G networks or as a mechanism for localising a service to networks where QoS demands are not met, as an attempt to improve their performance.

7.3 Future Work

The work in this thesis highlights a few areas that are subject to future research. This section proposes future work and work currently in progress in the fields of QoS and Traffic Monitoring, Datacentre Selection and Security.

7.3.1 QoS and Traffic Monitoring

The developed solution for QoS and traffic monitoring assumes that this data is made available when needed by establishing probe connections or taking samples of the network traffic from the network interfaces of the Datacentre. However, the process of gathering this data upon initiation of the mechanisms poses a delay to the execution time. An improved solution to acquiring this information is by taking samples at frequent intervals and maintaining a record of the network utilisation and QoS. Ideally this should be done on the client-side in order to avoid centralising this process to a datacentre that serves thousands of users. A rudimentary script in PowerShell has already been developed for sampling network latency at frequent intervals. The user can input the latency allowance as well as set a threshold to the amount of packets observed over the latency allowance. Such scripts are commonly used within datacentres to monitor the availability of hosts and network conditions. When the threshold is exceeded, the script moves the VM to an alternative host defined by the user. This script can be used as the basis for the development of a more complex mechanism.

A mechanism that monitors network QoS through latency measurements is currently in development for the SP. This mechanism sends echo packets at frequent intervals and maintains a record of the average roundtrip time. The recorded latency, along with the packet size can be used to estimate the characteristics of a connection in terms of bandwidth, roundtrip time and jitter. This information can then be used to calculate the service rate of a network and use it as input to the proposed QoS model. Furthermore, in order to maintain up-to-date information on the performance of third-party networks that a user might join along their path, a separate service is proposed that will use either SP or Internet Control Message Protocol (ICMP) echo packets to probe the network conditions of remote datacentres and networks thus providing information that can be used to model the QoS along a user's entire path. With these mechanisms gathering network metrics at frequent intervals, the required information to determine the time and target of a service migration will be available at the moment that the service requires it and negate the need for the service to include these mechanisms. This will result in a shorter execution time which will allow the service to respond more quickly to user mobility and network changes. Ideally, network performance information could be retrieved directly from the network using SDN technology, thus negating the need for performing any measurements on the client or server side, however, this solution implies that all the networks in the Internet use the same SDN solution and co-operate in providing this information to the end-users and service providers.

It is also possible to modify the SP to provide information on actual network throughput generated by the service. This calculation can also be performed on the client-side along with the QoS monitoring. Alternatively, samples of network throughput can be taken at frequent intervals from the network card of the client device using the host operating system. However, this presents a less accurate measurement method because some of the traffic may not be related to the service in question. The prototype presented in this thesis only runs a single service and therefore there is no other traffic that might skew the results, however in a real datacentre environment or client device, there may be multiple services using a single network interface. It would be more accurate to perform the measurement at the transport layer where it is possible to distinguish which service generates the network traffic. The development of this mechanism for the Simple Protocol has been proposed and is planned for future implementation. The aim is to maintain a log on the client device which describes the current usage as well as historic data that may be used to determine usage patterns at different times of the day and help identifying when a service should be localised.

7.3.2 Resolving Datacentre Locations

One of the challenges emerging from this thesis is the identification of datacentres that peer with the user's network attachment. While many datacentres may peer with many networks, this does not necessarily mean that QoS will be equal in every scenario. When localising a service, the first information required is the user's network ID. The Autonomous System Number (ASN) is the term used to describe the unique ID for each network on the Internet and it can be derived by querying online services created for this purpose (Team-Cymru, 2014) [64]. Using the ASN, it is possible to identify peering relationships between networks but it first requires building a database that contains all the ASNs on the Internet, as well as their peering relationships. In addition, the database should contain datacentre IDs that peer with each network. This information is not readily available but it can be constructed using information from online public databases. The challenge is to build a mechanism that looks up the user's ASN and determines peering datacentres. The peering datacentres serve as potential targets for the migration. By gathering QoS metrics between each of the peering datacentre's and the user's ASN, it is possible to determine the best host for the service. For this purpose, a PowerShell script was created as a side-project and as a template for future work. The script identifies the user's public IP address, and resolves it to the ASN of the user's current network. This information can then be used in conjunction with the aforementioned peering database to identify target datacentres and request QoS data from each one.

7.3.3 Security

Although the security specifics of migrating services using Cloud Interoperability mechanisms fall outside the scope of this thesis and are currently explored by Cloud Interoperability researchers, there are several other aspects of the proposed system that present potential security weaknesses. The main security concern is that of ensuring that performance data for an individual client and service are exchanged in a way that prevents impersonation or tampering. Establishing and securing a telemetry channel between the client device and the service is a necessity for preventing malicious users from feeding false data into the system and triggering false migrations. There are several different attacks that can be used against the system involving false prevention or triggering of a migration, tampering with the destination of the migration by giving false user location and overloading a Cloud by a co-ordinated attack that forces services to migrate to it. To prevent such attacks there are three proposed directions for future research. The first direction is to secure the communication channel between the device and the service so that man-in-the-middle attacks cannot be used to alter the performance data. The second direction is to validate the mechanisms that handle this data on the client and Cloud side so that a malicious programmer will not alter them and tamper with the data at the point where it is gathered and processed. Finally, security mechanisms on the Cloud should prevent any services from migrating in or out without first establishing an agreement between the source and destination Clouds. This is partially covered in the proposed service delivery framework as part of the migration layer which receives information from the Service Management layer to establish which services can move depending on their requirements and the capabilities of the target Cloud.

7.3.4 Scalability

The prototype solution proposed in Chapter 5 does not take into account scalability problems that will arise when the mechanisms are deployed in large-scale networks with a large number of users. Although problems such as migration time and throughput measurement become apparent, the exact approach to collecting this information and calculating in advance whether or not a migration is possible, can only be identified via simulation. The particular areas of interest are the calculation of throughput between Cloud hosts which defines the migration time of a service, the collection of throughput metrics between services and clients/Cloud back-end and the monitoring of user mobility. Centralising data collection to the Cloud will add to the cost of deploying services because there will be additional processing that has to be done on the Cloud but has the advantage of releasing resources from the thin-client and removing the need to transmit such telemetry over the network. The alternative approach of calculating throughput on the client, may affect the performance and battery life of a mobile device and it will also add to the network traffic between the client and the Cloud. In either approach, it should be up to the service to gather its own telemetry data rather than implement a global mechanism on each Cloud that performs the measurements and reports them to the applications. The main argument behind this approach is that ultimately, services will have to be decoupled from any particular

Cloud provider and should be able to make their own decisions on when and where to move. A simulation of such a solution deployed in a large scale network with thousands of clients will offer insights into how much traffic is generated by telemetry information, what is the added computation load on Clouds and services and in the case of hundreds or thousands of concurrent migrations, what level of bandwidth is required between participating datacentres. Similarly, the approach proposed in Chapter 6 for measuring the overall QoS of a connection over multiple networks, a simulation on a large scale network will reveal what the additional bandwidth requirements are for probing remote networks to report on their QoS, how much bandwidth it will cost the network to transmit telemetry information between clients and services and how the system would perform under a scenario of hundreds of concurrent migrations. Lastly, user mobility detection and reporting will require the deployment of additional network mechanisms that can track a device and report its movement. Furthermore, a global map that describes the coverage of networks as well as their peering relationships with datacentres will also need to be used as reference for the system to find the appropriate location for services. From a QoS perspective, a service migration is more complex in terms of finding a target datacentre because a local datacentre may not always provide the best performance. Therefore, it would be necessary for a client to query a list of datacentres that are internal or external to their network and then inform the service on which one offers the best performance. Once again, these mechanisms will have to be defined and tested in a simulation in order to gain a better understanding of the complexity and consequences on the Internet.

7.3.5 Saleability

If scalability issues are addressed for the mechanisms proposed in this thesis, the deployment of these solutions can assist in delivering MCC services more efficiently and with consistent performance. Cloud providers will gain a business opportunity in localising their datacentres and acquiring additional sources of incomes from service providers whose services use resources from different Clouds as they migrate according to user mobility. Furthermore, network and service providers can create additional streams of revenue by employing new charging models for customers who wish higher levels of performance. Finally, users will gain the ability to customise the performance of their Cloud applications according to their personal requirements in order to reduce their cost or maximise their performance.

7.4 Closing Statement

In conclusion, this thesis has demonstrated the need to take into account user mobility and service traffic characteristics for the purposes of traffic management and QoS provisioning in the future Internet. The proposed solutions for managing traffic and QoS on a per-user basis answer the research question and also highlight the need for real-time dynamic monitoring of network utilisation and QoS for each user. The development of personalised Cloud services, in tandem with Cloud Interoperability standards and faster network access technologies, constitute the ground for facilitating traffic localisation and QoS management through the use of service portability. The focus for the future Internet is on the real-time dynamic adaptation of networks based on traffic requirements and user mobility. The work presented in this thesis is a small step towards developing some of the required mechanisms.

Bibliography

1. Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A, Buyya, R. 2014. *Cloud-Based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges*. Communications Surveys & Tutorials, IEEE, vol.16, no.1, pp.337-368.
2. Alexander, K. 2012. *Fat Client Game Streaming or Cloud Gaming.* [online] Available at: https://blogs.akamai.com/2012/08/part-2-fat-client-game-streaming-or-cloud-gaming.html [Accessed: 08/08/2014]
3. Alexander, K. 2012. *The Future of Cloud Gaming is Still Cloudy*. [online] Available at: https://blogs.akamai.com/2012/08/the-future-of-cloud-gaming-isstill-cloudy.html [Accessed: 08/08/2014]
4. Bahl, P., Han, R. Y., Li, L. E., & Satyanarayanan, M. 2012. *Advancing the state of mobile cloud computing.* In the Proceedings of the third ACM workshop on Mobile cloud computing and services pp. 21-28.
5. Berg, R. V. D. 2008. *How the 'Net' works: an introduction to peering and transit*. [online] Available at: http://arstechnica.com/features/2008/09/peering-and-transit/ [Accessed: 06/08/2014].
6. CAIDA. (2014). *IPv4 and IPv6 AS Core: Visualizing IPv4 and IPv6 Internet Topology at a Macroscopic Scale*. Center for Applied Internet Data Analysis [online] Available at: http://www.caida.org/research/topology/as_core_network [Accessed: 06/08/2014].
7. Chen, K. T., Chang, Y. C., Tseng, P. H., Huang, C. Y., & Lei, C. L. (2011, November). *Measuring the latency of cloud gaming systems.* In Proceedings of the 19th ACM international conference on Multimedia (pp. 1269-1272). ACM.
8. Chen, P. M., & Noble, B. D. 2001. *When virtual is better than real (operating system relocation to virtual machines)*. In the Proceedings of the Eighth Workshop on Hot Topics in Operating Systems, pp. 133-138.
9. Cheng, L., Jean, K., Ocampo, R., & Galis, A. 2006. *Service-aware Overlay Adaptation in Ambient Networks*. IEEE International Multi-Conference on Computing in the Global Information Technology, Bucharest, Romania, pp. 21.
10. CISCO. (2014). Unified Data Center Fabric: Reduce Costs and Improve Flexibility. [online] Available at: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-5020-switch/white_paper_c11-462181.html [Accessed: 06/08/2014].
11. Clinch, S., Harkes, J., Friday, A., Davies, N., Satyanarayanan, M. 2012. *How close is close enough? Understanding the role of cloudlets in supporting display appropriation by mobile users*. IEEE International Conference on Pervasive Computing and Communications (PerCom), pp.122-127.
12. Cook Network, Consultants. 2002. *Changing Role of Peering & Transit in IP Network Interconnection Economics*. [online] Available at: http://cookreport.com/newsletter-sp-542240406/pdf?download=135:pdf-back-issues&start=120 [Accessed: 06/08/2014].
13. Cox, L. P., Murray, C. D., & Noble, B. D. 2002. *Pastiche: Making backup cheap and easy*. ACM SIGOPS Operating Systems Review, vol.36, pp.285-298.
14. Darmois, E. 2013. *"Cloud Standards Coordination" Final Report v1.0.* [online] Available at: http://csc.etsi.org/Application/documentApp/documentinfo/?documentId=204&fromList=Y [Accessed: 06/08/2014]

15. Dinh, T. H., Lee, C., Niyato, D. & Wang, P. 2011. *A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches*. Wireless Communications and Mobile Computing, Wiley, pp.1–38.
16. Duan, Q., Yuhong Y., & Vasilakos, A. V. 2012. *A Survey on Service-Oriented Network Virtualization Toward Convergence of Networking and Cloud Computing*. IEEE Transactions on Network and Service Management, vol.9, no.4, pp.373-392.
17. Erl, T. 2005. *Service-Oriented Architecture – Concepts, Technology, and Design*. Prentice Hall.
18. Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., & Estrin, D. 2010. *Diversity in smartphone usage*. In Proceedings of the 8th international conference on Mobile systems, applications, and services, pp. 179-194.
19. Filippi, P. 2014. *It's Time to Take Mesh Networks Seriously (And Not Just for the Reasons You Think)*. [online] Available at: http://www.wired.com/2014/01/its-time-to-take-mesh-networks-seriously-and-not-just-for-the-reasons-you-think [Accessed: 06/08/2014]
20. Finn, A. 2013. *Windows Server 2012 R2 Hyper-V Live Migration Improvements*. [online] Available at: http://www.aidanfinn.com/?p=14907 [Accessed: 06/08/2014]
21. Ford, A., Raiciu, C., Handley, M., Bonaventure, O. *TCP Extensions for Multipath Operation with Multiple Addresses*. [online] Available at: http://tools.ietf.org/html/rfc6824 [Accessed: 15/08/2014].
22. Gonzalez, M. C., Hidalgo, C. A., & Barabasi, A. L. (2008). *Understanding individual human mobility patterns*. Nature, vol., 453, no., 7196 pp. 779-782.
23. Gu, X., Nahrstedt, K., Chang, N., & Ward, C. 2003. *QoS-Assured Service Composition in Managed Service Overlay Networks*. 23rd International Conference on Distributed Computing Systems, Providence, Rhode Island, USA, pp.194-201.
24. Gu, X., Nahrstedt, K., Messer, A. Greenberg, I., Milojicic, D., 2004. *Adaptive offloading for pervasive computing*. Pervasive Computing, IEEE, vol.3, no.3, pp.66-73.
25. Harney, E., Goasguen, S., Martin, J., Murphy, M., & Westall, M. 2007. *The Efficacy of Live Virtual Machine Migrations over the Internet*. In Proceedings of the 3rd VTDC.
26. Hobfeld, T., Hausheer, D., Hecht, F. V., Lehrieder, F., Oechsner, S., Papafili, I., & Stiller, B. 2009. *An Economic Traffic Management Approach to Enable the TripleWin for Users, ISPs, and Overlay Providers*. In Future Internet Assembly pp. 24-34.
27. Hobfeld, T.; Schatz, R.; Varela, M.; Timmerer, C., 2012. *Challenges of QoE management for cloud applications*. Communications Magazine, IEEE, vol.50, no.4, pp.28-36.
28. Huang, D. 2011. *Mobile Cloud Computing*. IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter, vol.6, no.10, pp.27-31.
29. Huston, G. 1999. Web caching. *The Internet Protocol Journal, Vol. 2, Issue* 3, pp. 2-20. Also available online at: http://www.cisco.com/web/about/ac123/ac147/ac174/ac199/about_cisco_ipj_archive_article09186a00800c8903.html [Accessed 09/08/2014].
30. Hwang, K., Dongarra, J., & Fox, G. C. 2012. *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. Elsevier Science, pp. 160-161, 163-165.
31. Ibrahim, K. Z., Hofmeyr, S., Iancu, C., & Roman, E. 2011. *Optimized pre-copy live migration for memory intensive applications.* At the Proceedings of 2011

International Conference for High Performance Computing, Networking, Storage and Analysis, no.40.

32. IEEE Cloud Computing. 2013. *IEEE Intercloud Testbed An Open, Global, Cloud Interoperability Project.* [online] Available at: http://cloudcomputing.ieee.org/intercloud#sthash.AUB3acEU.dpuf . [Accessed: 06/08/2014].
33. IEEE Cloud Computing. 2013. IEEE *Intercloud testbed project announces founding members*. [online] Available at: http://cloudcomputing.ieee.org/intercloud/press-release [Accessed: 06/08/2014]
34. ITU. 2007. Recommendation P.10: New Appendix I – Definition of Quality of Experience (QoE) [online] Available at: http://www.itu.int/rec/T-REC-P.10-200701-S!Amd1 [Accessed: 06/02/2015]
35. ITU. 2008. Recommendation E.800: Terms and definitions related to quality of service and network performance including dependability. [online] Available at: http://www.itu.int/rec/T-REC-E.800/en [Accessed: 06/02/2015]
36. Krishnan, R., Madhyastha, H. V., Srinivasan, S., Jain, S., Krishnamurthy, A., Anderson, T., & Gao, J. 2009. *Moving beyond end-to-end path information to optimize CDN performance*. In the Proceedings of the 9th ACM SIGCOMM conference on Internet measurement, pp. 190-201.
37. Labovitz, C., Iekel-Johnson, S., McPherson, D., Oberheide, J., & Jahanian, F. 2011. *Internet inter-domain traffic*. ACM SIGCOMM Computer Communication Review, vol.41, no.4, pp.75-86.
38. Leadbetter, R. 2010. *In Theory: Is this how OnLive works?* [online] Available at: http://www.eurogamer.net/articles/digitalfoundry-onlive-beta-article [Accessed: 09/08/2014].
39. Lee, K., Yoon, H., & Park, S. 2013. *A Service Path Selection and Adaptation Algorithm in Service-Oriented Network Virtualization Architecture*. International Conference on Parallel and Distributed Systems (ICPADS), pp.516-521.
40. Liu, H., Jin, H., Liao, X., Hu, L., & Yu, C. 2009. *Live migration of virtual machine based on full system trace and replay.* In the Proceedings of the 18th ACM international symposium on High performance distributed computing, pp.101-110.
41. Liu, H., Xu, Y., & Zeng, Q. A. 2006. *Modelling and performance analysis of a channel reservation handoff scheme for multimedia wireless and mobile networks using smart antennas*. International Journal of High Performance Computing and Networking, pp.13–22.
42. Mapp, G. E., Shaikh, F., Cottingham, D., Crowcroft, J., & Baliosian, J. 2007. *Y-Comm: a global architecture for heterogeneous networking.* In the Proceedings of the 3rd international conference on Wireless Internet, no.22.
43. Mapp, G., and Cottingham, D., Shaikh, F., Vidales, P., Patanapongpibul, L., Baliosian, J., & Crowcroft, J. 2006. *An architectural framework for heterogeneous networking*. In the Proceedings of the International Conference on Wireless Information Networks and Systems.
44. Mapp, G., Katsriku, F., Aiash, M., Chinnam, N., Lopes, R., Moreira, E., Augusto, M. *2012*. *Exploiting location and contextual information to develop a comprehensive framework for proactive handover in heterogeneous environments*. Journal of Computer Networks and Communications, vol.2012, no. 748163.

45. Markoff, J. 2008. *Internet Traffic Begins to Bypass the U.S.* [online] Available at: http://www.nytimes.com/2008/08/30/business/30pipes.html?pagewanted=all&_r=1& [Accessed: 08/08/2014]
46. Mell, P., Grace, T. 2011. The NIST Definition of Cloud Computing [online] Available at: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf [Accessed: 06/02/2015]
47. Microsoft 2010. *Interoperability Elements of Cloud Platform*. [online] Available at: http://www.microsoft.com/cloud/interop/ [Accessed: 06/08/2014].
48. Microsoft. 2012. *Virtual Machine Live Migration Overview*. [online] Available at: http://technet.microsoft.com/en-us/library/hh831435.aspx [Accessed: 09/08/2014]
49. Nolle, T. 2013. *Three models of SDN explained*. [online] Available at: http://searchtelecom.techtarget.com/tip/Three-models-of-SDN-explained [Accessed: 15/08/2014].
50. Ocampo, R., Cheng, L., Lai, Z., & Galis, A. 2005. *ContextWare support for Network and service composition and Self-adaptation*. 2nd International Workshop on Mobility Aware Technologies and Applications, pp.84-95.
51. OnLive Team. 2013. *Fun Facts* [online] Available at: http://www.onlive.com/careers/fun_facts [Accessed: 09/08/2014].
52. OnLive Team. 2014. *Computer Display Resolution and the OnLive Game Service*. [online] Available at: https://support.onlive.com/hc/en-us/articles/201229150-Computer-Display-Resolution-and-the-OnLive-Game-Service [Accessed: 08/08/2014]
53. Piatek, M., Madhyastha, H. V., John, J. P., Krishnamurthy, A., & Anderson, T. E. 2009. *Pitfalls for ISP-friendly P2P design*. In Eighth ACM Worskhop on Hot Topics in Networks (HotNets).
54. Rhoton, J. 2009. C*loud Computing Explained: Implementation Handbook For Enterprises*. Recursive Press, pp.21.
55. Richardson, T., Bennett, F., Mapp, G. & Hopper, A. 1994. *A ubiquitous, personalized computing environment for all: Teleporting in an X Window System Environment*. Personal Communications, IEEE, vol.1, no.3, pp.6.
56. Riley, L. and Mapp, G. 2014. yRFC3: The Simple Protocol Lite (SP-Lite) Specification [online] Available at: http://www.mdx.ac.uk/__data/assets/pdf_file/0003/176592/yrfc3-SP-Lite.pdf [Accessed: 30/03/2015]
57. Sapuntzakis, C. P., Chandra, R., Pfaff, B., Chow, J., Lam, M. S., & Rosenblum, M. 2002. *Optimizing the migration of virtual computers*. ACM SIGOPS Operating Systems Review, vol.36, pp.377-390.
58. Sardis, F., Mapp, G., Loo, J., & Aiash, M., 2014. *Dynamic Edge-Caching for Mobile Users: Minimising Inter-AS traffic by Moving Cloud Services and VMs*. Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on, pp.144-149.
59. Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. 2009. *The case for VM-based cloudlets in mobile computing*. Pervasive Computing, IEEE, vol.8, no.4, pp.14-23.
60. Seibert, J., Torres, R., Mellia, M., Munafo, M. M., Nita-Rotaru, C., & Rao, S. 2012. *The Internet-Wide Impact of P2P Traffic Localization on ISP Profitability*. IEEE/ACM Transactions on Networking, vol.20, no.6, pp.1910-1923.
61. Shea, R., Jiangchuan L., Ngai, E.C., & Yong, C. 2013. *Cloud gaming: architecture and performance*. Network, IEEE, vol.27, no.4, pp.16-21.

62. Softperfect, 2014. *Connection Emulator* [Computer Programme] Available at: http://www.softperfect.com/products/connectionemulator/ [Accessed: 08/08/2014]
63. Su, Y. Y., & Flinn, J. 2005. *Slingshot: deploying stateful services in wireless hotspots*. In Proceedings of the 3rd international conference on Mobile systems, applications, and services, pp. 79-92.
64. Team Cymru, 2014. *Internet Security Research and Insight*. [online] Available at http://www.team-cymru.com/ [Accessed: 06/08/2014]
65. Tolia, N., Harkes, J., Kozuch, M., & Satyanarayanan, M. 2004. *Integrating Portable and Distributed Storage*. In FAST, Vol. 4, pp. 227-238.
66. Valancius, V., Lumezanu, C., Feamster, N., Johari, R., & Vazirani, V. V. 2011. *How many tiers?: pricing in the internet transit market*. *ACM SIGCOMM Computer Communication Review, vol.41, no.*4, pp.194-205.
67. Verbelen, T., Stevens, T., Simoens, P., De Turck, F., & Dhoedt, B. 2011. *Dynamic deployment and quality adaptation for mobile augmented reality applications*. Journal of Systems and Software, vol.84, no.11, pp.1871-1882.
68. Vij, D. K. & Bernstein, D. 2012. *Draft Standard for Intercloud Interoperability and Federation* (SSIF). [online] Available at: https://www.oasis-open.org/committees/download.php/46205/p2302-12-0002-00-DRFT-intercloud-p2302-draft-0-2.pdf [Accessed: 06/08/2014].
69. Warner, A. S., & Karman, F. A. 2010. *Defining the Mobile Cloud*. NASA I.T. Summit. [Presentation] Available at: www.nasa.gov/ppt/482352main_2010_Monday_1_Warner.Steven_r5.ppt [Accessed 15/08/2014]
70. Wen, Z. Y., & Hsiao, H. F. (2014, September). QoE-driven performance analysis of cloud gaming services. In Multimedia Signal Processing (MMSP), 2014 IEEE 16th International Workshop on (pp. 1-6). IEEE.
71. Woodcock, B. 2003. Internet Topology and Economics: *How Supply and Demand Influence the Changing Shape of the Global Network*. [Lecture] Available at: http://www.pch.net/resources/papers/topology-and-economics/Topology-and-Economics.ppt [Accessed 15/08/2014]
72. Zhu, W., Luo, C., Wang, J., & Li, S. 2011. *Multimedia cloud computing*. Signal Processing Magazine, IEEE, vol.28, no.3, pp.59-69.

Appendix A

Reactive Script Code

```
#Find local host name. Used later to find if the VHD is local or remote.

$hostname = hostname

#Set VM name.

$vmname = "client" #read-Host "What is the VM's name?"

#Find the VHD location for the VM. Used later to find if the VHD is local or
remote.

$getvm = Get-vm -name $vmname | Select -ExpandProperty HardDrives

$vhdloc = $getvm.Path

#Check the VM RAM size. First we have to enable VM resource metering which is off
by default.

Enable-VMResourceMetering -VMName $vmname

$vmstats = Get-VM -Name $vmname

[int]$vm = (($vmstats.memoryassigned/1024)/1024)

#Set estimated migration throughput.

[int]$mthr = read-host "What is the migration throughput? (MB/s)"

#Calculate the estimated migration time.

[int]$mtime = [int]$vm / [int]$mthr

"--------------------"

"VM size: $vm MB"

"Estimated Migration Time: $mtime seconds"

"--------------------"

#Set the user's estimated NDT.

[int]$ndt = read-host "What is the user's NDT? (seconds)"

#Check if there is enough time to migrate. If not, abort.

if ([int]$ndt -le [int]$mtime) {"Migration time for the VM is greater than NDT.
Aborting."}

else {"Sufficient NDT for migration. Calculating migration parameters."

     #Initialise varibles for counting RDC and VHD throughput.

    [int]$count = 0

    [int]$totalrdc = 0
    [int]$totalvhd = 0
#Measure RDC and VHD when VHD is at the same location as the VM
if ($vhdloc -match $hostname){"VHD local, measuring VHD throughput from Hyper-V counters"
#Take measurements of throughput for RDC and VHD.
    while ($count -lt 20){
        #Select NIC. We select bytesSent from the NIC because that represents data sent to RDC as opposed to received data which could be web activity.
        $interface = Get-WmiObject -class Win32_PerfFormattedData_Tcpip_NetworkInterface |select Name, BytesSentPersec |where {$_.Name -eq "Realtek PCIe GBE Family Controller"}
        [int]$rdcactivity = $interface.BytesSentPersec
        #Measure VHD from Hyper-V Counters
        $write = get-counter '\\host0\hyper-v virtual storage device(--?-unc-host0.latency.local-public-documents-hyper-v-virtual hard disks-rfx.vhdx)\write bytes/sec'
        [int]$writevalue = $write.CounterSamples[0].CookedValue
        $read = get-counter '\\host0\hyper-v virtual storage device(--?-unc-host0.latency.local-public-documents-hyper-v-virtual hard disks-rfx.vhdx)\read bytes/sec'
        [int]$readvalue = $read.CounterSamples[0].CookedValue
        #Sum of read/write operations
        [int]$vhdactivity = $readvalue + $writevalue
        #Total RDC and VHD activity
        [int]$totalrdc = [int]$totalrdc + $rdcactivity
        [int]$totalvhd = [int]$totalvhd + $vhdactivity
        Write-Host "RDC Throughput: $rdcactivity B/s"
        Write-Host "VHD Throughput: $vhdactivity B/s"
        $count++
        #We don't want to issue too many WMI queries too quickly on a production machine so we wait a while before sampling again
        Start-Sleep -milliseconds 300
        }
      #Average RDC and VHD traffic in MB/s
    [single]$rdc = "{0:N2}" -f ((([int]$totalrdc / 1024) / 1024) / [int]$count)
    [single]$vhd = "{0:N2}" -f ((([int]$totalvhd / 1024) / 1024) / [int]$count)

    "--------------------"

    "RDC Avg: $rdc MB/s"

    "VHD Avg: $vhd MB/s"

    "--------------------"

    if ([single]$rdc -le [single]$vhd) {"RDC is equal to or less than VHD. Or VM
inactive. No migration possible"}

    if ([single]$rdc -gt [single]$vhd) {[int]$ndtmin = ([int]$vm + ([single]$rdc *
[int]$mtime) + ([single]$vhd * ([int]$ndt - [int]$mtime))) / [single]$rdc

    "Min NDT: $ndtmin seconds"

        if ([int]$ndt -gt [int]$ndtmin) {"Migrating the VM to the user's
location."}

        else {"No sufficient NDT for traffic savings at current usage."}

        }

    }

#Measure RDC and VHD when VHD is in a remote location to the VM

If ($vhdloc -notmatch $hostname) {"VHD in remote location, measuring VHD throughput
from backbone NIC"

    #Take 10 measurements of throughput for RDC and VHD.

    while ($count -lt 20){

        #Select NIC. We select BytesSent from the NIC because that represents data
sent to RDC as opposed to received data which could be web activity.

        $interface = Get-WmiObject -class
Win32_PerfFormattedData_Tcpip_NetworkInterface |select Name, BytesSentPersec |where
{$_.Name -eq "Qualcomm Atheros AR8152 PCI-E Fast Ethernet Controller [NDIS 6.30]"}

        [int]$rdcactivity = [double]$interface.BytesSentPersec

        #Select NIC for VHD. We select BytesTotal because we are interested in VHD
read/write

        $interface2 = Get-WmiObject -class
Win32_PerfFormattedData_Tcpip_NetworkInterface |select Name, BytesTotalPersec|where
{$_.Name -eq "Realtek PCI GBE Family Controller"}

        [int]$vhdactivity = $interface2.BytesTotalPersec

        [int]$totalrdc = [int]$totalrdc + $rdcactivity

        [int]$totalvhd = [int]$totalvhd + $vhdactivity

        Write-Host "RDC Throughput: $rdcactivity B/s"

        Write-Host "VHD Throughput: $vhdactivity B/s"
        $count++

        #We don't want to issue too many WMI queries too quickly on a production
machine so we wait a while before sampling again

        Start-Sleep -milliseconds 300

        }

      #Average RDC and VHD traffic.

    [single]$rdc = "{0:N2}" -f (([int]$totalrdc / 1024) / 1024) / [int]$count

    [single]$vhd = "{0:N2}" -f (([int]$totalvhd / 1024) / 1024) / [int]$count

    "--------------------"

    "RDC Avg: $rdc MB/s"

    "VHD Avg: $vhd MB/s"

    "--------------------"

    #If RDC = VHD = 0, VM inactive.

    if ($rdc -eq 0 -and $vhd -eq 0) {"VM inactive. Aborting."}

    #If RDC less than or equal to VHD and both larger than 0.

    elseif  ($rdc  -le  $vhd)  {[int]$ndtmin  =  ([int]$vm  +  ([single]$vhd  *
[int]$mtime)) / [single]$vhd


                                  "Min NDT: $ndtmin seconds"


        if ($ndt -gt $ndtmin) {"Defaulting the VM to eliminate VHD traffic."}

        else {"No sufficient NDT for traffic savings at current usage."}

        }

    elseif  ($rdc  -gt  $vhd)  {[int]$ndtmin  =  ([int]$vm  +  ([single]$rdc  *
[int]$mtime)) / [single]$rdc


                  "Min NDT: $ndtmin seconds"


        if ($ndt -gt $ndtmin) {"Migrating the VM to the user's location."}

        else {"No sufficient NDT for traffic savings at current usage."}

        }

    }

}
```

Pre-emptive Script Code

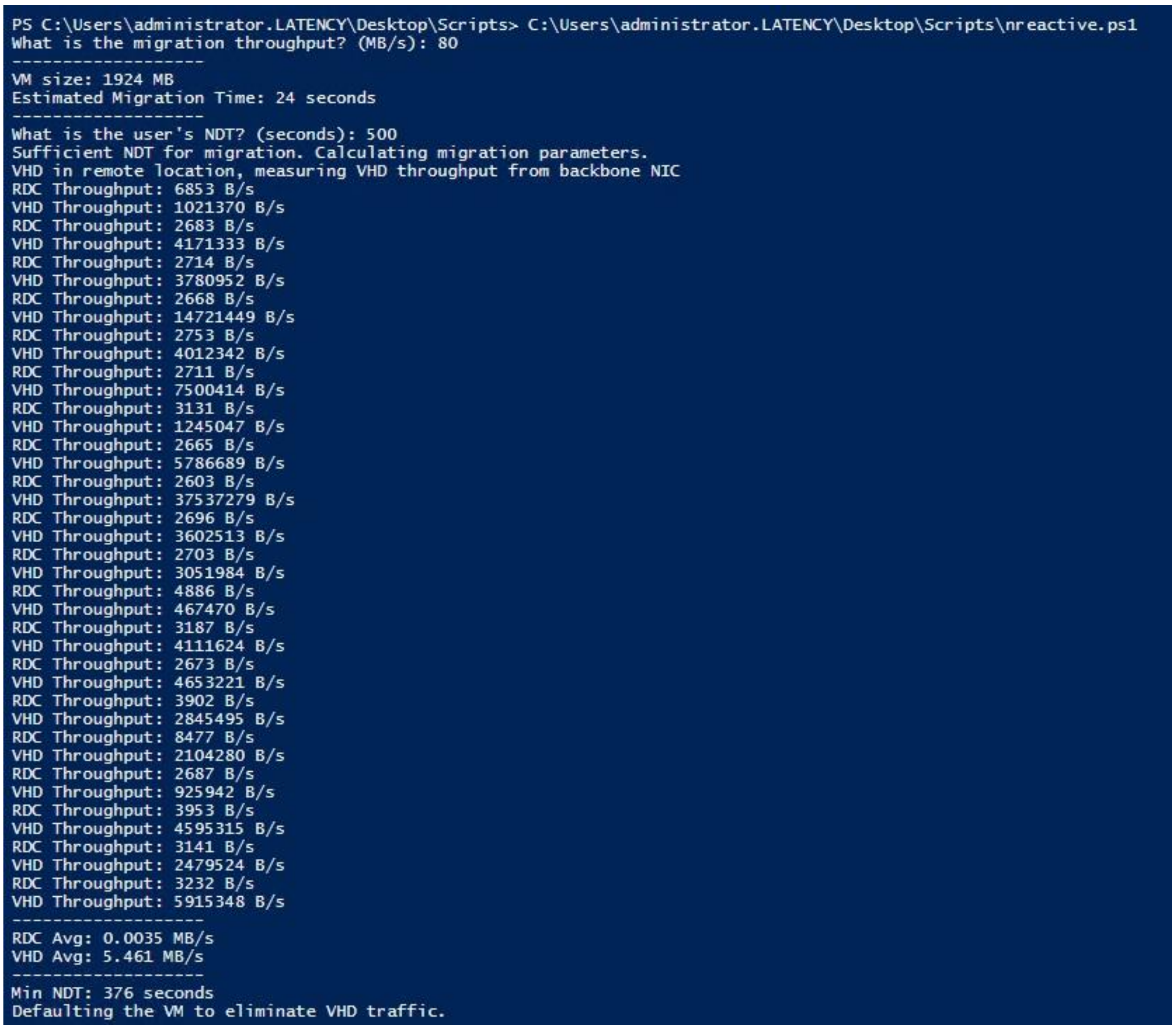
```
#Get name of VM host

[string]$hostname = hostname

#Table of NDT reported by the client in order of path priority. We calculate to sum
of NDT for later use.

$net=@{"Net-A" = 100 ; "Net-B" = 800 ; "Net-C" = 600 ; "Net-D" = 150}

[int]$ndttot = 0

$ndtvalues = $net.GetEnumerator() | Select Value

[int]$ndttot = ($ndtvalues | Measure-Object 'Value' -Sum).Sum

"Total NDT in current path is $ndttot"

#Table of hosts connected to each network.

$hosts=@{"Net-A" = "Host-A" ; "Net-B" = "Host-B" ; "Net-C" = "Host-C" ; "Net-D" =
"Host-D"}

#Set VM name.

$vmname = "client" #read-Host "What is the VM's name?"

#Set estimated migration throughput.

[int]$mthr = read-host "what is the migration throughput? (MB/s)"

#Check the VHD location for the VM.

$getvm = Get-vm -name $vmname | Select -ExpandProperty HardDrives

$vhdloc = $getvm.Path

#Check the VM RAM size and convert to MB. First we have to enable VM resource
metering which is off by default.

Enable-VMResourceMetering -VMName $vmname

$vmstats = Get-VM -Name $vmname

[int]$vm = (($vmstats.memoryassigned/1024)/1024)

#Calculate the estimated migration time.

[int]$mtime = [int]$vm / [int]$mthr

"--------------------"

"VM size: $vm MB"

"Estimated Migration Time: $mtime seconds"

"--------------------"

#Check if enough ndttot to migrate. if not abort.

if ([int]$ndttot -le [int]$mtime) {"Migration time for the VM is greater than total
```

```
NDT. Aborting."}

else {

    #Initialise varibles for counting RDC and VHD throughput.

    [int]$count = 0

    [int]$totalrdc = 0

    [int]$totalvhd = 0

#Measure RDC and VHD when VHD is at the same location as the VM

if ($vhdloc -match $hostname){"VHD local, measuring VHD throughput from Hyper-V
counters"

    #Take 20 measurements of throughput for RDC and VHD.

    while ($count -lt 20){

    #Select NIC. We select bytesSent from the NIC because that represents data sent
to RDC as opposed to received data which could be web activity.

        $interface = Get-WmiObject -class
Win32_PerfFormattedData_Tcpip_NetworkInterface |select Name, BytesSentPersec |where
{$_.Name -eq "Realtek PCIe GBE Family Controller"}

        [int]$rdcactivity = $interface.BytesSentPersec

       #Measure VHD from Hyper-V Counters

        $write = get-counter '\\host0\hyper-v virtual storage device(--?-unc-
host0.latency.local-public-documents-hyper-v-virtual hard disks-rfx.vhdx)\write
bytes/sec'

        [int]$writevalue = $write.CounterSamples[0].CookedValue


        $read = get-counter '\\host0\hyper-v virtual storage device(--?-unc-
host0.latency.local-public-documents-hyper-v-virtual hard disks-rfx.vhdx)\read
bytes/sec'

        [int]$readvalue = $read.CounterSamples[0].CookedValue

        #Sum of read/write operations

        [int]$vhdactivity = $readvalue + $writevalue

        #Total RDC and VHD activity

        [int]$totalrdc = [int]$totalrdc + $rdcactivity

        [int]$totalvhd = [int]$totalvhd + $vhdactivity

        Write-Host "RDC Throughput: $rdcactivity B/s"

        Write-Host "VHD Throughput: $vhdactivity B/s"

        $count++
        #We don't want to issue too many WMI queries too quickly on a production
machine so we wait a while before sampling again

        Start-Sleep -milliseconds 300

        }

   #Average RDC and VHD traffic in MB/s

    [single]$rdc = "{0:N2}" -f ((([int]$totalrdc / 1024) / 1024) / [int]$count)

    [single]$vhd = "{0:N2}" -f ((([int]$totalvhd / 1024) / 1024) / [int]$count)

    "-------------------"

    "RDC Avg: $rdc MB/s"

    "VHD Avg: $vhd MB/s"

    "-------------------"

    if ($rdc -le $vhd){"RDC is equal to or less than VHD. Or VM inactive. Do
nothing."}

    #If VHD local and RDC>VHD, calculate minimum NDT and look for target:

    if ($rdc -gt $vhd) {[int]$ndtmin = ([int]$vm + ([single]$rdc * [int]$mtime) +
([single]$vhd * ([int]$ndt - [int]$mtime))) / [single]$rdc

    "Min NDT: $ndtmin seconds"

        $newnet = $net.GetEnumerator() | select Name, Value |where  {$_.Value -ge
$ndtmin} |Sort-Object Name |Select -First 1

        $tloc = $newnet.Name

        $filterhost = $hosts.GetEnumerator() |Select Name, Value |where {$_.Name -
like $tloc}

        $thost = $filterhost.Value

        #If no target is found, abort.

        if ($thost -like "") {"No valid target network found for migration.
Aborting."}

       else {"Migrating to $thost."}


    }

}

#Measure RDC and VHD when VHD is in a remote location to the VM

If ($vhdloc -notmatch $hostname) {"VHD in remote location, measuring VHD throughput
from backbone NIC"

    #Take 10 measurements of throughput for RDC and VHD.

    while ($count -lt 20){
        #Select NIC. We select BytesSent from the NIC because that represents data
sent to RDC as opposed to received data which could be web activity.

        $interface                =                Get-WmiObject                -class
Win32_PerfFormattedData_Tcpip_NetworkInterface |select Name, BytesSentPersec |where
{$_.Name -eq "Qualcomm Atheros AR8152 PCI-E Fast Ethernet Controller [NDIS 6.30]"}

        [int]$rdcactivity = $interface.BytesSentPersec

      #Select NIC for VHD. We select BytesTotal because we are interested in VHD
read/write

        $interface2               =                Get-WmiObject                -class
Win32_PerfFormattedData_Tcpip_NetworkInterface |select Name, BytesTotalPersec|where
{$_.Name -eq "Realtek PCI GBE Family Controller"}

        [int]$vhdactivity = $interface2.BytesTotalPersec

        [int]$totalrdc = [int]$totalrdc + $rdcactivity

        [int]$totalvhd = [int]$totalvhd + $vhdactivity

        Write-Host "RDC Throughput: $rdcactivity B/s"

        Write-Host "VHD Throughput: $vhdactivity B/s"

       $count++

        #We don't want to issue too many WMI queries too quickly on a production
machine so we wait a while before sampling again

        Start-Sleep -milliseconds 300

        }

    #Average RDC and VHD traffic.

    [single]$rdc = "{0:N2}" -f ((([int]$totalrdc / 1024) / 1024) / [int]$count)

    [single]$vhd = "{0:N2}" -f ((([int]$totalvhd / 1024) / 1024) / [int]$count)

    "-------------------"

    "RDC Avg: $rdc MB/s"

    "VHD Avg: $vhd MB/s"

    "-------------------"

    #If RDC = VHD = 0, VM inactive.

    if ($rdc -eq 0 -and $vhd -eq 0) {Write-Host "VM inactive. Aborting."}

    #If RDC greater than VHD

    elseif ($rdc -gt $vhd) {

        #calculate minimum NDT for eliminating RDC and for eliminating VHD. We need
to know both to decide what to do.

        [int]$ndtmin = ([int]$vm + ([single]$rdc * [int]$mtime)) / [single]$rdc
        [int]$ndtmin2 = ([int]$vm + ([single]$vhd * [int]$mtime)) / [single]$vhd

        $newnet = $net.GetEnumerator() | select Name, Value |where  {$_.Value -ge
$ndtmin} |Sort-Object Name |Select -First 1

        $tloc = $newnet.Name

        $filterhost = $hosts.GetEnumerator() |Select Name, Value |where {$_.Name -
like $tloc}

        $thost = $filterhost.Value

        #If no target is found based on minimum NDT for RDC reduction, examine if
total NDT is sufficient for VHD reduction instead.

        if  ($thost  -like  ""  -and  $ndttot  -gt  $ndtmin2)  {"No  target  found  for
migration. Eliminating VHD traffic instead. Returning VM to default location."}

        elseif($thost  -like""  -and  $ndttot  -le  $ndtmin2)  {"No  target  found  for
migration. Not possible to eliminate VHD traffic. Aborting."}

        else {"Migrating to $thost."}

        }

    #If RDC less or equal to VHD try to default the VM to eliminate VHD.

    elseif  ($rdc  -le  $vhd)  {[int]$ndtmin  =  ([int]$vm  +  ([single]$vhd  *
[int]$mtime)) / [single]$vhd

                                  "Min NDT: $ndtmin seconds"

              if ($ndttot -gt $ndtmin) {"Defaulting the VM."}

           else {"Do nothing."}

      }

}

}
```

ASN-Resolution Script Code

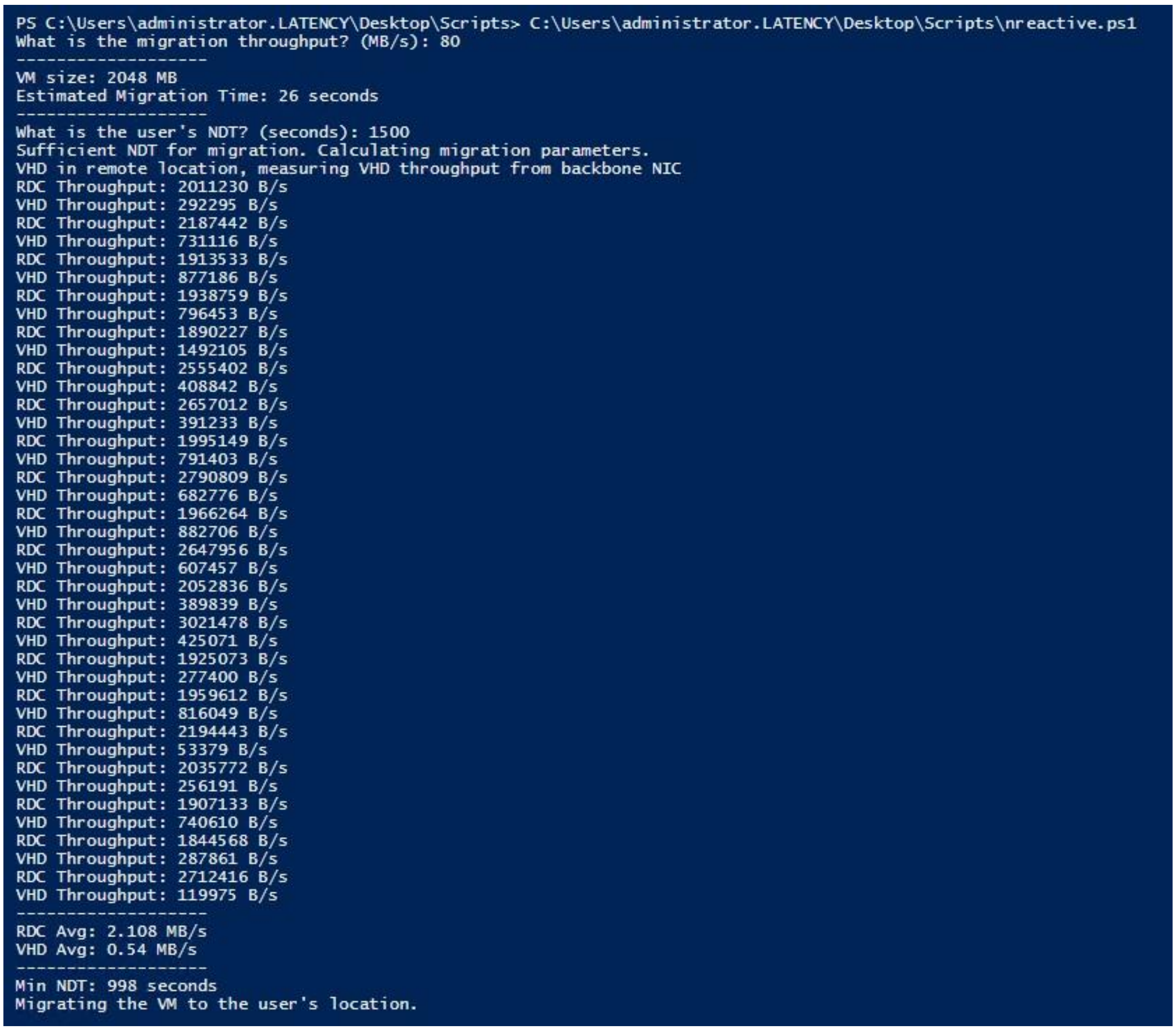
```
$IPAddress = Invoke-WebRequest ifconfig.me/ip

$IPAddressParts = $IPAddress.Content.Split('.')

$RIPAddress                                                                          =
"$($IPAddressParts[3].Trim()).$($IPAddressParts[2]).$($IPAddressParts[1]).$($IPAddr
essParts[0])"

##Services we're getting the ASN information from.

$OriginService= ".origin.asn.cymru.com"

$ASNService = ".asn.cymru.com"

##Resolve the name against the ASN-IP service

$NameToResolve = "$RIPAddress" + "$OriginService"

try{

        $DNSRecords = Resolve-DnsName $NameToResolve  -Type TXT -ErrorAction Stop

   }

    catch{

        throw "Could not find AS information for $NameToResolve"

        exit

    }

foreach ($Record in $DNSRecords)

{

    $Result = New-Object System.Object

    $Result | Add-Member -Type NoteProperty -Name IPAddress -Value  $IPAddress

        $Result    |    Add-Member    -Type    NoteProperty    -Name    ASNumber    -Value
$Record.Strings.Split("|")[0].Trim()

    $Result    |    Add-Member    -Type    NoteProperty    -Name    ASPrefix    -Value
$Record.Strings.Split("|")[1].Trim()

    $Result    |    Add-Member    -Type    NoteProperty    -Name    Locale    -Value
$Record.Strings.Split("|")[2].Trim()

    $NameToResolve = "AS" +$Result.ASNumber + $ASNService


    try{

        $DNSRecords = Resolve-DnsName $NameToResolve  -Type TXT

       }
    catch{

        throw "Could not resolve detailed infromation for AS" +$Result.ASNumber

        exit

    }

    $Result    |    Add-Member    -Type    NoteProperty    -Name    Description    -Value
$DNSRecords.Strings.Split("|")[4].Trim()

    $Result

}
```

Traffic Management Prototype Screenshots



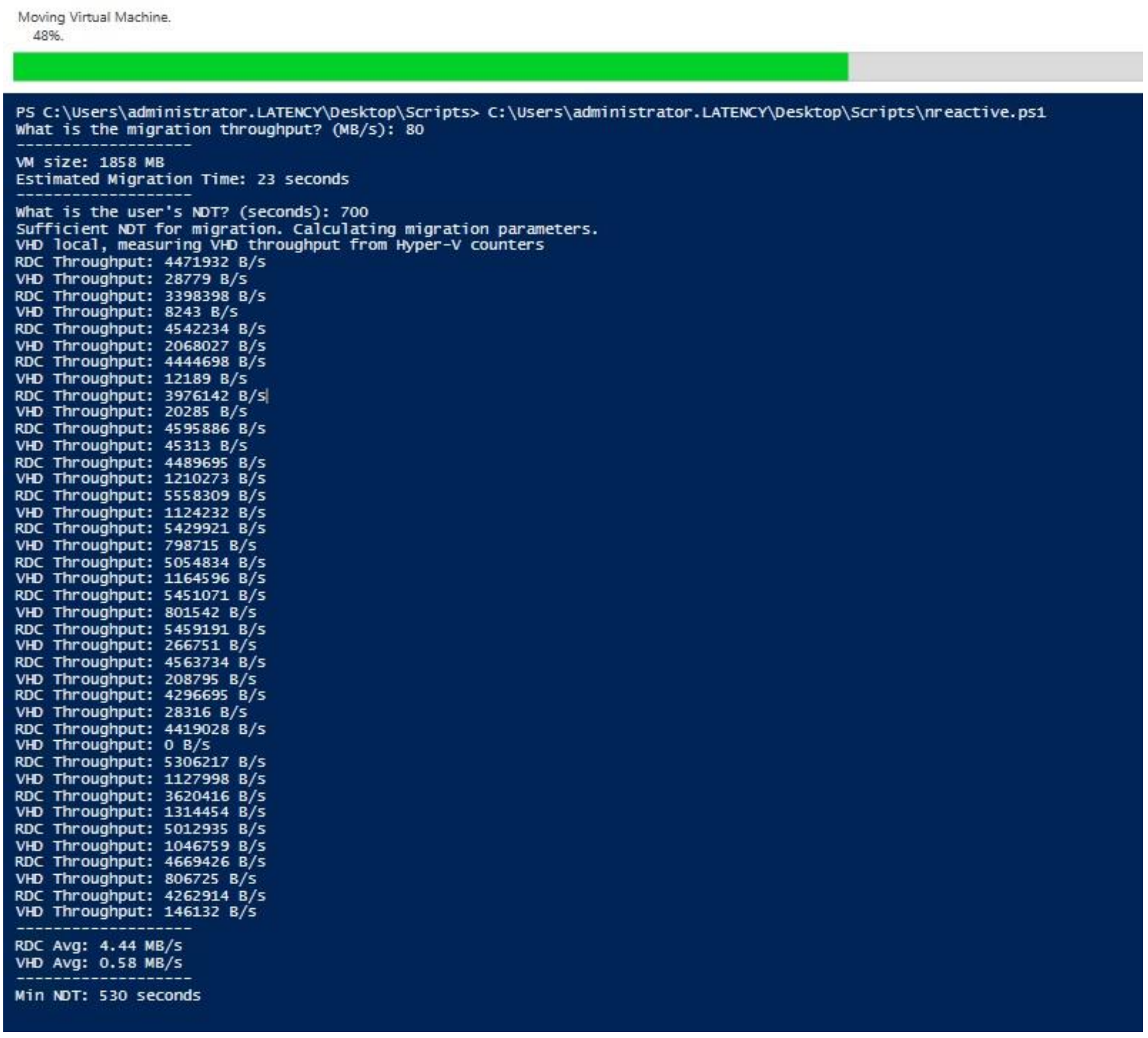
```
PS C:\Users\administrator.LATENCY\Desktop\Scripts> C:\Users\administrator.LATENCY\Desktop\Scripts\nreactive.ps1
What is the migration throughput? (MB/s): 80
-------------------
VM size: 1924 MB
Estimated Migration Time: 24 seconds
-------------------
What is the user's NDT? (seconds): 500
Sufficient NDT for migration. Calculating migration parameters.
VHD in remote location, measuring VHD throughput from backbone NIC
RDC Throughput: 6853 B/s
VHD Throughput: 1021370 B/s
RDC Throughput: 2683 B/s
VHD Throughput: 4171333 B/s
RDC Throughput: 2714 B/s
VHD Throughput: 3780952 B/s
RDC Throughput: 2668 B/s
VHD Throughput: 14721449 B/s
RDC Throughput: 2753 B/s
VHD Throughput: 4012342 B/s
RDC Throughput: 2711 B/s
VHD Throughput: 7500414 B/s
RDC Throughput: 3131 B/s
VHD Throughput: 1245047 B/s
RDC Throughput: 2665 B/s
VHD Throughput: 5786689 B/s
RDC Throughput: 2603 B/s
VHD Throughput: 37537279 B/s
RDC Throughput: 2696 B/s
VHD Throughput: 3602513 B/s
RDC Throughput: 2703 B/s
VHD Throughput: 3051984 B/s
RDC Throughput: 4886 B/s
VHD Throughput: 467470 B/s
RDC Throughput: 3187 B/s
VHD Throughput: 4111624 B/s
RDC Throughput: 2673 B/s
VHD Throughput: 4653221 B/s
RDC Throughput: 3902 B/s
VHD Throughput: 2845495 B/s
RDC Throughput: 8477 B/s
VHD Throughput: 2104280 B/s
RDC Throughput: 2687 B/s
VHD Throughput: 925942 B/s
RDC Throughput: 3953 B/s
VHD Throughput: 4595315 B/s
RDC Throughput: 3141 B/s
VHD Throughput: 2479524 B/s
RDC Throughput: 3232 B/s
VHD Throughput: 5915348 B/s
-------------------
RDC Avg: 0.0035 MB/s
VHD Avg: 5.461 MB/s
-------------------
Min NDT: 376 seconds
Defaulting the VM to eliminate VHD traffic.
```

Screenshot 1 Migration from remote location to the home location using the Reactive script



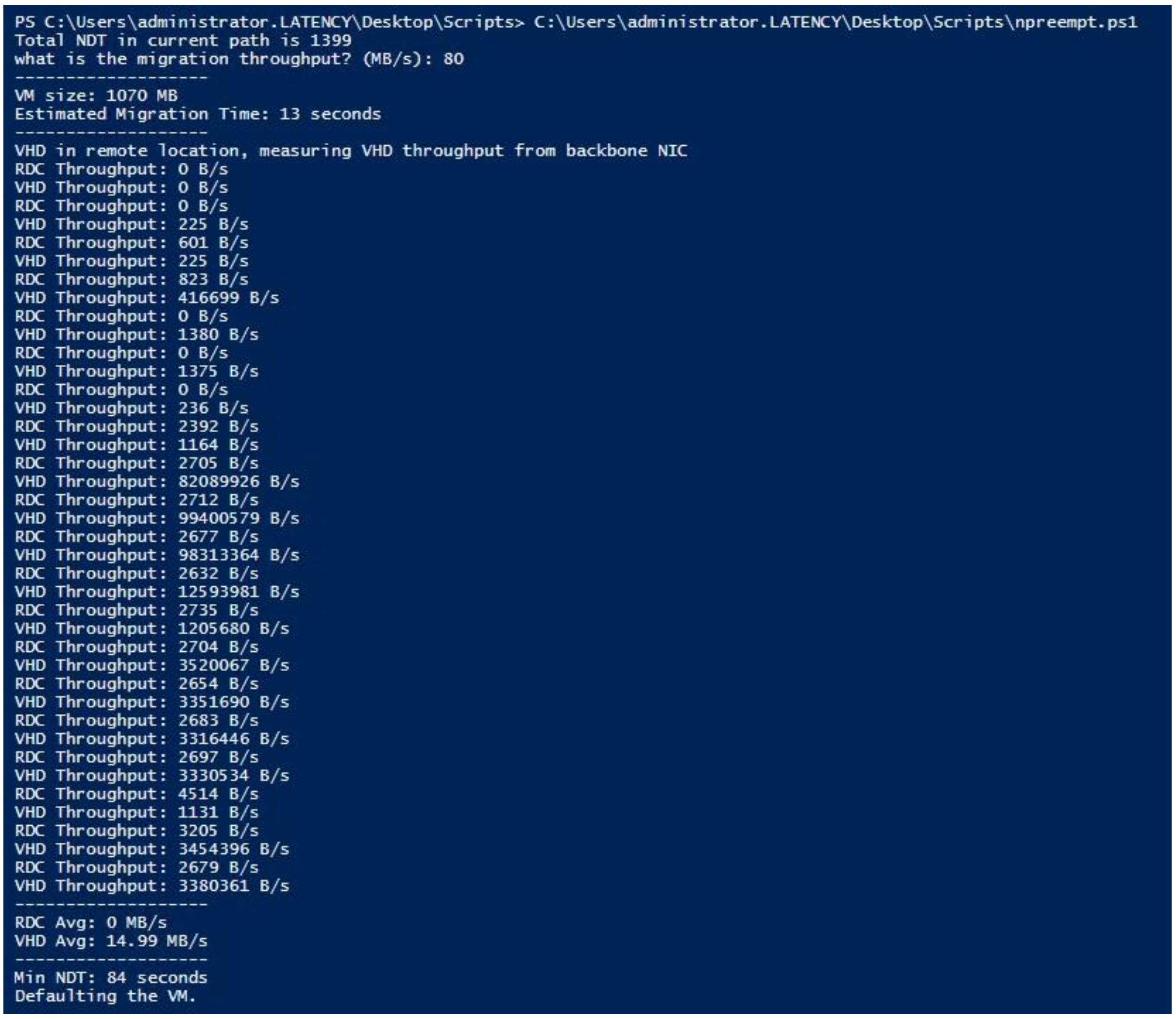
```
PS C:\Users\administrator.LATENCY\Desktop\Scripts> C:\Users\administrator.LATENCY\Desktop\Scripts\nreactive.ps1
What is the migration throughput? (MB/s): 80
-------------------
VM size: 2048 MB
Estimated Migration Time: 26 seconds
-------------------
What is the user's NDT? (seconds): 1500
Sufficient NDT for migration. Calculating migration parameters.
VHD in remote location, measuring VHD throughput from backbone NIC
RDC Throughput: 2011230 B/s
VHD Throughput: 292295 B/s
RDC Throughput: 2187442 B/s
VHD Throughput: 731116 B/s
RDC Throughput: 1913533 B/s
VHD Throughput: 877186 B/s
RDC Throughput: 1938759 B/s
VHD Throughput: 796453 B/s
RDC Throughput: 1890227 B/s
VHD Throughput: 1492105 B/s
RDC Throughput: 2555402 B/s
VHD Throughput: 408842 B/s
RDC Throughput: 2657012 B/s
VHD Throughput: 391233 B/s
RDC Throughput: 1995149 B/s
VHD Throughput: 791403 B/s
RDC Throughput: 2790809 B/s
VHD Throughput: 682776 B/s
RDC Throughput: 1966264 B/s
VHD Throughput: 882706 B/s
RDC Throughput: 2647956 B/s
VHD Throughput: 607457 B/s
RDC Throughput: 2052836 B/s
VHD Throughput: 389839 B/s
RDC Throughput: 3021478 B/s
VHD Throughput: 425071 B/s
RDC Throughput: 1925073 B/s
VHD Throughput: 277400 B/s
RDC Throughput: 1959612 B/s
VHD Throughput: 816049 B/s
RDC Throughput: 2194443 B/s
VHD Throughput: 53379 B/s
RDC Throughput: 2035772 B/s
VHD Throughput: 256191 B/s
RDC Throughput: 1907133 B/s
VHD Throughput: 740610 B/s
RDC Throughput: 1844568 B/s
VHD Throughput: 287861 B/s
RDC Throughput: 2712416 B/s
VHD Throughput: 119975 B/s
-------------------
RDC Avg: 2.108 MB/s
VHD Avg: 0.54 MB/s
-------------------
Min NDT: 998 seconds
Migrating the VM to the user's location.
```

Screenshot 2 Migration from remote location to user’s new location using the Reactive script



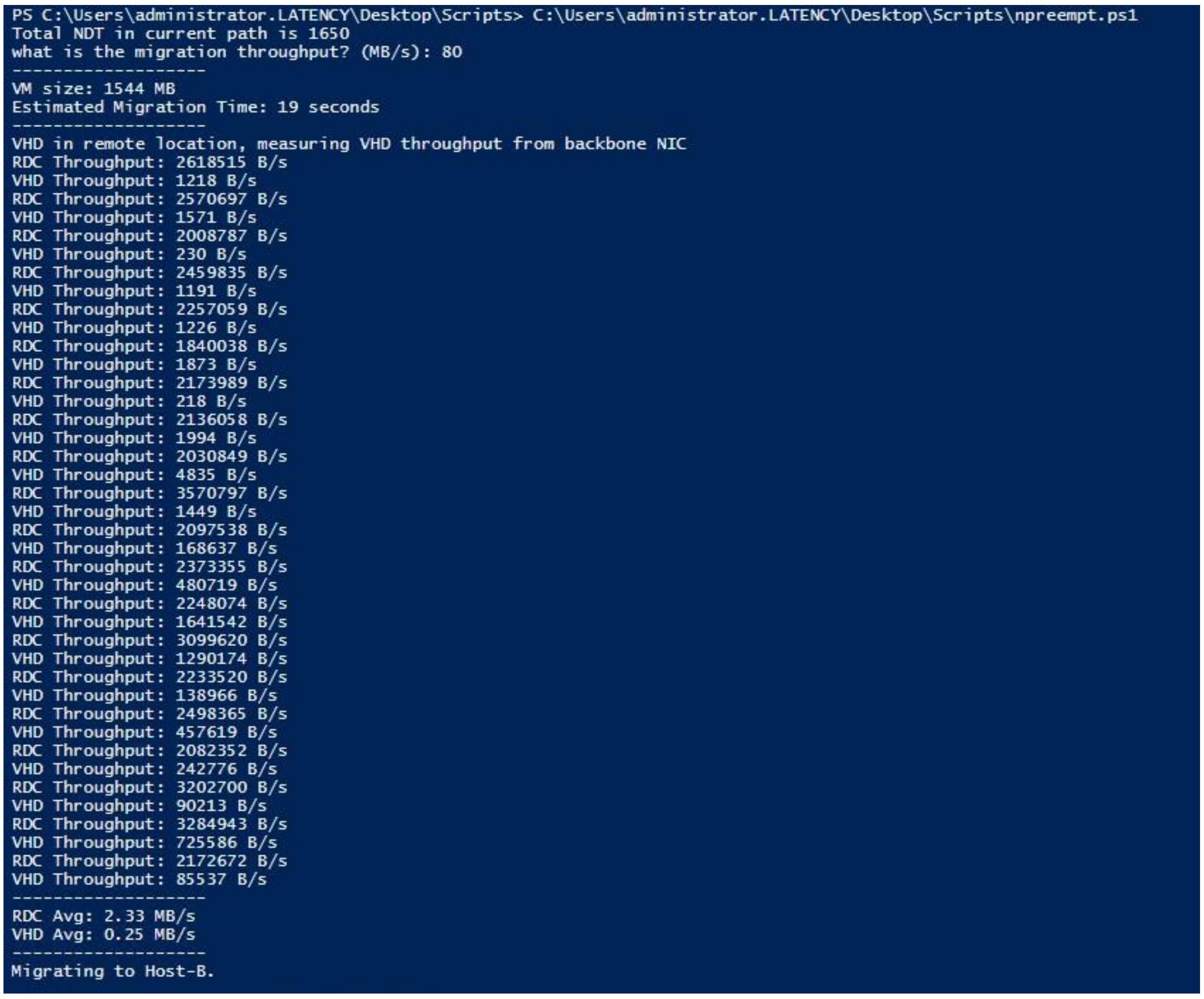
```
PS C:\Users\administrator.LATENCY\Desktop\Scripts> C:\Users\administrator.LATENCY\Desktop\Scripts\nreactive.ps1
What is the migration throughput? (MB/s): 80
-------------------
VM size: 1858 MB
Estimated Migration Time: 23 seconds
-------------------
What is the user's NDT? (seconds): 700
Sufficient NDT for migration. Calculating migration parameters.
VHD local, measuring VHD throughput from Hyper-V counters
RDC Throughput: 4471932 B/s
VHD Throughput: 28779 B/s
RDC Throughput: 3398398 B/s
VHD Throughput: 8243 B/s
RDC Throughput: 4542234 B/s
VHD Throughput: 2068027 B/s
RDC Throughput: 4444698 B/s
VHD Throughput: 12189 B/s
RDC Throughput: 3976142 B/s
VHD Throughput: 20285 B/s
RDC Throughput: 4595886 B/s
VHD Throughput: 45313 B/s
RDC Throughput: 4489695 B/s
VHD Throughput: 1210273 B/s
RDC Throughput: 5558309 B/s
VHD Throughput: 1124232 B/s
RDC Throughput: 5429921 B/s
VHD Throughput: 798715 B/s
RDC Throughput: 5054834 B/s
VHD Throughput: 1164596 B/s
RDC Throughput: 5451071 B/s
VHD Throughput: 801542 B/s
RDC Throughput: 5459191 B/s
VHD Throughput: 266751 B/s
RDC Throughput: 4563734 B/s
VHD Throughput: 208795 B/s
RDC Throughput: 4296695 B/s
VHD Throughput: 28316 B/s
RDC Throughput: 4419028 B/s
VHD Throughput: 0 B/s
RDC Throughput: 5306217 B/s
VHD Throughput: 1127998 B/s
RDC Throughput: 3620416 B/s
VHD Throughput: 1314454 B/s
RDC Throughput: 5012935 B/s
VHD Throughput: 1046759 B/s
RDC Throughput: 4669426 B/s
VHD Throughput: 806725 B/s
RDC Throughput: 4262914 B/s
VHD Throughput: 146132 B/s
-------------------
RDC Avg: 4.44 MB/s
VHD Avg: 0.58 MB/s
-------------------
Min NDT: 530 seconds
```

Screenshot 3 VM migration from Home location to the user's current location using the Reactive Script



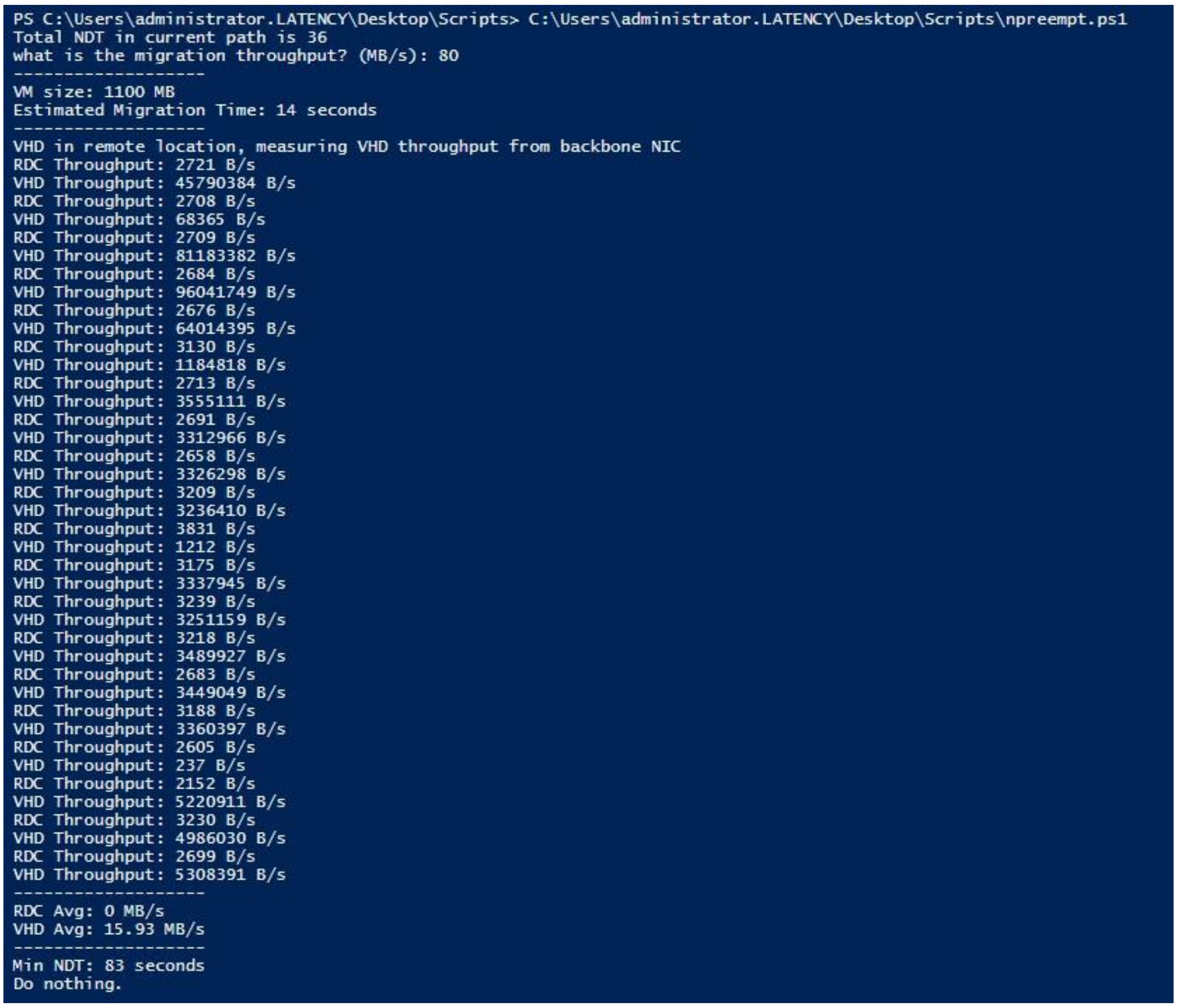
```
PS C:\Users\administrator.LATENCY\Desktop\Scripts> C:\Users\administrator.LATENCY\Desktop\Scripts\npreempt.ps1
Total NDT in current path is 1399
what is the migration throughput? (MB/s): 80
-------------------
VM size: 1070 MB
Estimated Migration Time: 13 seconds
-------------------
VHD in remote location, measuring VHD throughput from backbone NIC
RDC Throughput: 0 B/s
VHD Throughput: 0 B/s
RDC Throughput: 0 B/s
VHD Throughput: 225 B/s
RDC Throughput: 601 B/s
VHD Throughput: 225 B/s
RDC Throughput: 823 B/s
VHD Throughput: 416699 B/s
RDC Throughput: 0 B/s
VHD Throughput: 1380 B/s
RDC Throughput: 0 B/s
VHD Throughput: 1375 B/s
RDC Throughput: 0 B/s
VHD Throughput: 236 B/s
RDC Throughput: 2392 B/s
VHD Throughput: 1164 B/s
RDC Throughput: 2705 B/s
VHD Throughput: 82089926 B/s
RDC Throughput: 2712 B/s
VHD Throughput: 99400579 B/s
RDC Throughput: 2677 B/s
VHD Throughput: 98313364 B/s
RDC Throughput: 2632 B/s
VHD Throughput: 12593981 B/s
RDC Throughput: 2735 B/s
VHD Throughput: 1205680 B/s
RDC Throughput: 2704 B/s
VHD Throughput: 3520067 B/s
RDC Throughput: 2654 B/s
VHD Throughput: 3351690 B/s
RDC Throughput: 2683 B/s
VHD Throughput: 3316446 B/s
RDC Throughput: 2697 B/s
VHD Throughput: 3330534 B/s
RDC Throughput: 4514 B/s
VHD Throughput: 1131 B/s
RDC Throughput: 3205 B/s
VHD Throughput: 3454396 B/s
RDC Throughput: 2679 B/s
VHD Throughput: 3380361 B/s
-------------------
RDC Avg: 0 MB/s
VHD Avg: 14.99 MB/s
-------------------
Min NDT: 84 seconds
Defaulting the VM.
```

Screenshot 4 VM returned to Home location using the Pre-Emptive script.



```
PS C:\Users\administrator.LATENCY\Desktop\Scripts> C:\Users\administrator.LATENCY\Desktop\Scripts\npreempt.ps1
Total NDT in current path is 1650
what is the migration throughput? (MB/s): 80
-------------------
VM size: 1544 MB
Estimated Migration Time: 19 seconds
-------------------
VHD in remote location, measuring VHD throughput from backbone NIC
RDC Throughput: 2618515 B/s
VHD Throughput: 1218 B/s
RDC Throughput: 2570697 B/s
VHD Throughput: 1571 B/s
RDC Throughput: 2008787 B/s
VHD Throughput: 230 B/s
RDC Throughput: 2459835 B/s
VHD Throughput: 1191 B/s
RDC Throughput: 2257059 B/s
VHD Throughput: 1226 B/s
RDC Throughput: 1840038 B/s
VHD Throughput: 1873 B/s
RDC Throughput: 2173989 B/s
VHD Throughput: 218 B/s
RDC Throughput: 2136058 B/s
VHD Throughput: 1994 B/s
RDC Throughput: 2030849 B/s
VHD Throughput: 4835 B/s
RDC Throughput: 3570797 B/s
VHD Throughput: 1449 B/s
RDC Throughput: 2097538 B/s
VHD Throughput: 168637 B/s
RDC Throughput: 2373355 B/s
VHD Throughput: 480719 B/s
RDC Throughput: 2248074 B/s
VHD Throughput: 1641542 B/s
RDC Throughput: 3099620 B/s
VHD Throughput: 1290174 B/s
RDC Throughput: 2233520 B/s
VHD Throughput: 138966 B/s
RDC Throughput: 2498365 B/s
VHD Throughput: 457619 B/s
RDC Throughput: 2082352 B/s
VHD Throughput: 242776 B/s
RDC Throughput: 3202700 B/s
VHD Throughput: 90213 B/s
RDC Throughput: 3284943 B/s
VHD Throughput: 725586 B/s
RDC Throughput: 2172672 B/s
VHD Throughput: 85537 B/s
-------------------
RDC Avg: 2.33 MB/s
VHD Avg: 0.25 MB/s
-------------------
Migrating to Host-B.
```

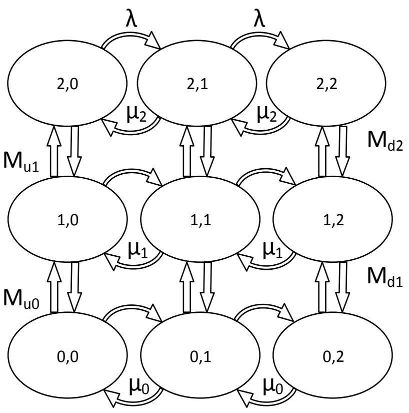
Screenshot 5 VM migrating from remote location to a target network using the Pre-emptive script.



```
PS C:\Users\administrator.LATENCY\Desktop\Scripts> C:\Users\administrator.LATENCY\Desktop\Scripts\npreempt.ps1
Total NDT in current path is 36
what is the migration throughput? (MB/s): 80
-------------------
VM size: 1100 MB
Estimated Migration Time: 14 seconds
-------------------
VHD in remote location, measuring VHD throughput from backbone NIC
RDC Throughput: 2721 B/s
VHD Throughput: 45790384 B/s
RDC Throughput: 2708 B/s
VHD Throughput: 68365 B/s
RDC Throughput: 2709 B/s
VHD Throughput: 81183382 B/s
RDC Throughput: 2684 B/s
VHD Throughput: 96041749 B/s
RDC Throughput: 2676 B/s
VHD Throughput: 64014395 B/s
RDC Throughput: 3130 B/s
VHD Throughput: 1184818 B/s
RDC Throughput: 2713 B/s
VHD Throughput: 3555111 B/s
RDC Throughput: 2691 B/s
VHD Throughput: 3312966 B/s
RDC Throughput: 2658 B/s
VHD Throughput: 3326298 B/s
RDC Throughput: 3209 B/s
VHD Throughput: 3236410 B/s
RDC Throughput: 3831 B/s
VHD Throughput: 1212 B/s
RDC Throughput: 3175 B/s
VHD Throughput: 3337945 B/s
RDC Throughput: 3239 B/s
VHD Throughput: 3251159 B/s
RDC Throughput: 3218 B/s
VHD Throughput: 3489927 B/s
RDC Throughput: 2683 B/s
VHD Throughput: 3449049 B/s
RDC Throughput: 3188 B/s
VHD Throughput: 3360397 B/s
RDC Throughput: 2605 B/s
VHD Throughput: 237 B/s
RDC Throughput: 2152 B/s
VHD Throughput: 5220911 B/s
RDC Throughput: 3230 B/s
VHD Throughput: 4986030 B/s
RDC Throughput: 2699 B/s
VHD Throughput: 5308391 B/s
-------------------
RDC Avg: 0 MB/s
VHD Avg: 15.93 MB/s
-------------------
Min NDT: 83 seconds
Do nothing.
```

Screenshot 6 Pre-emptive script aborting the migration process due to insufficient NDT.

Appendix B



3x3 Queueing Model Solution

$$(M_{u0}+\lambda)P_{0,0}=\mu_0 P_{0,1}+M_{d1}P_{1,0}$$

$$(M_{u0}+\lambda+\mu_0)P_{0,1}=M_{d1}P_{1,1}+\mu_0 P_{0,2}+\lambda P_{0,0}$$

$$(M_{u0}+\mu_0)P_{0,2}=M_{d1}P_{1,2}+\lambda P_{0,1}$$

$$(M_{u1}+M_{d1}+\lambda)P_{1,0}=M_{u0}P_{0,0}+M_{d2}P_{2,0}+\mu_1 P_{1,1}$$

$$(M_{u1}+M_{d1}+\lambda+\mu_1)P_{1,1}=M_{u0}P_{0,1}+M_{d2}P_{2,1}+\mu_1 P_{1,2}+\lambda P_{1,0}$$

$$(M_{u1}+M_{d1}+\mu_1)P_{1,2}=M_{u0}P_{0,2}+M_{d2}P_{2,2}+\lambda P_{1,1}$$

$$(M_{d2}+\lambda)P_{2,0}=M_{u1}P_{1,0}+\mu_2 P_{2,1}$$

$$(M_{d2}+\lambda+\mu_2)P_{2,1}=M_{u1}P_{1,1}+\mu_2 P_{2,2}+\lambda P_{2,0}$$

$$(M_{d2}+\mu_2)P_{2,2}=M_{u1}P_{1,2}+\lambda P_{2,1}$$

Starting from Chain 0:

$$P_{0,2}=a_{0,2}P_{1,2}+b_{0,2}P_{0,1} \qquad \frac{M_{d1}}{(M_{u0}+\mu_0)}=a_{0,2} \qquad \frac{\lambda}{(M_{u0}+\mu_0)}=b_{0,2}$$

$$P_{0,0}=a_{0,0}P_{0,1}+b_{0,0}P_{1,0} \qquad \frac{\mu_0}{(M_{u0}+\lambda)}=a_{0,0}\ \frac{M_{d1}}{(M_{u0}+\lambda)}=b_{0,0}$$

$$P_{0,1}=a_{0,1}P_{1,1}+b_{0,1}P_{1,2}+c_{0,1}P_{1,0}$$

$$\frac{M_{d1}}{(M_{u0}+\lambda+\mu_0-\mu_0 b_{0,2}-\lambda a_{0,0})}=a_{0,1}\frac{\mu_0 a_{0,2}}{(M_{u0}+\lambda+\mu_0-\mu_0 b_{0,2}-\lambda a_{0,0})}=b_{0,1}\frac{\lambda b_{0,0}}{(M_{u0}+\lambda+\mu_0-\mu_0 b_{0,2}-\lambda a_{0,0})}=c_{0,1}$$

$$P_{0,0}=a_{0,0}a_{0,1}P_{1,1}+a_{0,0}b_{0,1}P_{1,2}+\left(a_{0,0}c_{0,1}+b_{0,0}\right)P_{1,0}$$

$$P_{0,2}=\left(a_{0,2}+b_{0,2}b_{0,1}\right)P_{1,2}+b_{0,2}a_{0,1}P_{1,1}+b_{0,2}c_{0,1}P_{1,0}$$

Chain 2:

$$P_{2,2}=a_{2,2}P_{1,2}+b_{2,2}P_{2,1} \qquad \frac{M_{u1}}{(M_{d2}+\mu_2)}=a_{2,2} \qquad \frac{\lambda}{(M_{d2}+\mu_2)}=b_{2,2}$$

$$P_{2,0}=a_{2,0}P_{1,0}+b_{2,0}P_{2,1} \qquad \frac{M_{u1}}{(M_{d2}+\lambda)}=a_{2,0}\ \frac{\mu_2}{(M_{d2}+\lambda)}=b_{2,0}$$

$$P_{2,1}=a_{2,1}P_{1,1}+b_{2,1}P_{1,2}+c_{2,1}P_{1,0}$$

$\frac{M_{u1}}{(M_{d2}+\lambda+\mu_2-\lambda b_{2,0}-\mu_2 b_{2,2})} = a_{2,1} \frac{\mu_2 a_{2,2}}{(M_{d2}+\lambda+\mu_2-\lambda b_{2,0}-\mu_2 b_{2,2})} = b_{2,1} \frac{\lambda a_{2,0}}{(M_{d2}+\lambda+\mu_2-\lambda b_{2,0}-\mu_2 b_{2,2})} = c_{2,1}$

$P_{2,0} = (a_{2,0} + b_{2,0}c_{2,1})P_{1,0} + b_{2,0}a_{2,1}P_{1,1} + b_{2,0}b_{2,1}P_{1,2}$

$P_{2,2} = (a_{2,2} + b_{2,2}b_{2,1})P_{1,2} + b_{2,2}a_{2,1}P_{1,1} + b_{2,2}c_{2,1}P_{1,0}$

Chain 1:

$P_{1,2} = a_{1,2}P_{1,1} + b_{1,2}P_{1,0}$

$\frac{(M_{u0}b_{0,2}a_{0,1}+M_{d2}b_{2,2}a_{2,1}+\lambda)}{(M_{u1}+M_{d1}+\mu_1-M_{d2}a_{2,2}-M_{d2}b_{2,2}b_{2,1}-M_{u0}a_{0,2}-M_{u0}b_{0,2}b_{0,1})} = a_{1,2}$

$\frac{(M_{u0}b_{0,2}c_{0,1}+M_{d2}b_{2,2}c_{2,1})}{(M_{u1}+M_{d1}+\mu_1-M_{d2}a_{2,2}-M_{d2}b_{2,2}b_{2,1}-M_{u0}a_{0,2}-M_{u0}b_{0,2}b_{0,1})} = b_{1,2}$

Revisit previous chains to substitute P$_{1,2}$:

$P_{2,2} = (a_{2,2}a_{1,2} + b_{2,2}b_{2,1}a_{1,2} + b_{2,2}a_{2,1})P_{1,1} + (a_{2,2}b_{1,2} + b_{2,2}b_{2,1}b_{1,2} + b_{2,2}c_{2,1})P_{1,0}$

$P_{2,0} = (a_{2,0} + b_{2,0}c_{2,1} + b_{2,0}b_{2,1}b_{1,2})P_{1,0} + (b_{2,0}a_{2,1} + b_{2,0}b_{2,1}a_{1,2})P_{1,1}$

$P_{2,1} = (a_{2,1} + b_{2,1}a_{1,2})P_{1,1} + (b_{2,1}b_{1,2} + c_{2,1})P_{1,0}$

$P_{0,0} = (a_{0,0}a_{0,1} + a_{0,0}b_{0,1}a_{1,2})P_{1,1} + (a_{0,0}c_{0,1} + b_{0,0} + a_{0,0}b_{0,1}b_{1,2})P_{1,0}$

$P_{0,2} = (a_{0,2}a_{1,2} + b_{0,2}b_{0,1}a_{1,2} + b_{0,2}a_{0,1})P_{1,1} + (b_{0,2}b_{0,1}b_{1,2} + b_{0,2}c_{0,1} + a_{0,2}b_{1,2})P_{1,0}$

$P_{0,1} = (a_{0,1} + b_{0,1}a_{1,2})P_{1,1} + (c_{0,1} + b_{0,1}b_{1,2})P_{1,0}$

Back to chain 1:

$P_{1,0} = \mathrm{u}P_{1,1}$

$$\frac{(M_{u0}a_{0,0}a_{0,1}+M_{u0}a_{0,0}b_{0,1}a_{1,2}+\mu_1+M_{d2}b_{2,0}a_{2,1}+M_{d2}b_{2,0}b_{2,1}a_{1,2})}{(M_{u1}+M_{d1}+\lambda-M_{u0}a_{0,0}c_{0,1}-M_{u0}b_{0,0}-M_{u0}a_{0,0}b_{0,1}b_{1,2}-M_{d2}a_{2,0}-M_{d2}b_{2,0}c_{2,1}-M_{d2}b_{2,0}b_{2,1}b_{1,2})} = u$$

Back to previous probabilities and substitute for P$_{1,0}$:

$P_{2,2} = (a_{2,2}a_{1,2} + b_{2,2}b_{2,1}a_{1,2} + b_{2,2}a_{2,1} + a_{2,2}b_{1,2}\mathrm{u} + b_{2,2}b_{2,1}b_{1,2}\mathrm{u} + b_{2,2}c_{2,1}\mathrm{u})P_{1,1}$

$P_{2,0} = (b_{2,0}a_{2,1} + b_{2,0}b_{2,1}a_{1,2} + \mathrm{u}a_{2,0} + \mathrm{u}b_{2,0}c_{2,1} + \mathrm{u}b_{2,0}b_{2,1}b_{1,2})P_{1,1}$

$P_{2,1} = (a_{2,1} + b_{2,1}a_{1,2} + \mathrm{u}b_{2,1}b_{1,2} + \mathrm{u}c_{2,1})P_{1,1}$

$P_{0,0} = (a_{0,0}a_{0,1} + a_{0,0}b_{0,1}a_{1,2} + \mathrm{u}a_{0,0}c_{0,1} + \mathrm{u}b_{0,0} + \mathrm{u}a_{0,0}b_{0,1}b_{1,2})P_{1,1}$

$P_{0,2} = (a_{0,2}a_{1,2} + b_{0,2}b_{0,1}a_{1,2} + b_{0,2}a_{0,1} + b_{0,2}b_{0,1}b_{1,2}u + b_{0,2}c_{0,1}u + a_{0,2}b_{1,2}u)P_{1,1}$

$P_{0,1} = (a_{0,1} + b_{0,1}a_{1,2} + \mathrm{u}c_{0,1} + \mathrm{u}b_{0,1}b_{1,2})P_{1,1}$

$P_{1,2} = (a_{1,2} + b_{1,2}\mathrm{u})P_{1,1}$

Sum of all:

$1 = P_{0,0} + P_{0,1} + P_{0,2} + P_{1,0} + P_{1,1} + P_{1,2} + P_{2,0} + P_{2,1} + P_{2,2}$