ABSTRACT

BABAOGLU, AHMET CAN. Verification Services for the Choice-Based Internet of the Future. (Under the direction of Dr. Rudra Dutta and Dr. George Rouskas.)

The Internet has grown from its inception as a special-purpose internetwork into a general multi-purpose world-wide facility enabling education, commerce, governance, and societal communication, all in the space of a few decades. Over this time, and accelerating in the last decade or so, increasing demands and a growing variety of use cases are posing new challenges on the architecture prompting re-thinking and re-architecting of the network. One thread of research in such architectural considerations involves the issue of choice. The lack of alternative network services brings little economic incentive for the network service providers to make investments to deploy new technologies and improve the quality of their network services. In addition, most user flows goes through several providers, thus there is no effective mechanism in the current Internet to provide feedback to users about which provider is the cause of the performance problems they experience. One solution to these problems is to create a more competitive open market where providers can advertise their network services, and users can choose their desired set of network services to satisfy their needs. In this solution, the users have the option to choose another service if they are not satisfied. However, even in this solution, the root cause of the performance problems still can not be found and it brings us to the lack of a robust feedback capability.

In this work, we investigate a solution to this fundamental missing piece of the Internet, the measurement and verification capability of the network services offered in the Internet, that indirectly pushes more responsibility to the network providers to fulfill their requirements for high quality services. Our work, while rooted in standard expectations of economic theory, is not in economics itself. Rather, it is in defining, designing, and realizing architectural entities and interactions in technical terms that can realize verification services essential to enabling such economic interactions.

Our work is threefold; after giving a literature overview of the research on future Internet and Internet measurement, we first propose an architecture that defines the roles, interactions and design choices to enable a Choice-Based Verification Service. We then describe the results and analysis of a series of tests, which start with our work on measurement frameworks in wired and wireless environments and continue with the simulation, the mechanism introduced and the actual prototype of this work deployed into a real system, the GENI meso-scale testbed. Finally, we investigate and validate whether such informed choices with verification service actually lead to better overall results. We use energy-efficiency as a practical and useful domain for a case study and show the simulation results, which greatly increase the appeal of this work as applicable real-world network services.

Verification Services for the Choice-Based Internet of the Future

by
Ahmet Can Babaoglu

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2014

APPROVED BY:

DEDICATION

To my dear mother and father who provided me unconditional support and love.
To my grandmother and grandfather for always being inspiring figures for me.

BIOGRAPHY

Ahmet Can Babaoglu was born in Istanbul, Turkey. He is a PhD candidate in the Department of Computer Science at North Carolina State University. He has received his Bachelor of Science degree in Computer Engineering from Bogazici University in 2008 in Istanbul, Turkey.

He has joined North Carolina State University in 2008 and received his Master of Science degree in Computer Science from North Carolina State University in Raleigh, NC in 2011. He has received two outstanding graduate teaching assistantship awards during his teaching assistantship, in addition to his work as research assistant for several research projects. His research focuses upon Network Architecture Design and Network Measurement. He has given research presentations, live demonstrations and tutorials at several conferences.

He is proudly joining Riverbed Technology in the next step of his career.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Dr. Rudra Dutta for his help. He has always put trust on me and without his invaluable guidance and advices, I would have never acquired the skill set I currently possess. I also would like to express my gratitude to my co-advisor Dr. George Rouskas for his excellent advices and feedback. I also would like to thank my PhD committee members; Dr. David Thuente and Dr. Mihail Sichitiu for agreeing to be in my committee and for their constructive feedback. I also would like to thank Dr. Ilia Baldine for his support and Dr. Shu Huang for their feedback and valuable discussions. I also would like to thank NC State University Computer Science Department for always being supportive to me during my PhD, as well as NSF (National Science Foundation) for supporting my research.

I also would like to thank my friends for their support during my PhD; Dr. Attila Yavuz, Dr. Harun Demircioglu, Dr. Mustafa Berke Yelten, Nazli Dokuzoglu, Namik Kemal Temizer, Dr. Anjing Wang and Trisha Biswas for their support during my PhD.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Table 1: Abbreviations and Meaning

Chooser	The entity initiating a ChoiceNet Service
FIA	Future Internet Architecture
FIND	Future INternet Design
GENI	Global Environment for Network Innovations
IMF	Integrated Measurement Framework
IXP	Internet Exchange Point
MASP	Measurement Analysis Service Provider
MCSP	Measurement Composition Service Provider
MD	Measurement Data
MMCS	Measurement Metric Converter Service
MM	Measurement Mechanism
MP	Measurement Point
MPLS	Multi Protocol Label Switching
MSP	Measurement Service Provider
NS-3	Network Simulator 3
NSP	Network Service Provider
NSG	Network Service Gateway
OMF	Orbit Management Framework
OWD	One Way Delay
QoS	Quality of Service
PerfSONAR	Performance focused Service Oriented Network monitoring ARchitecture
SILO	Service Integrated ControL and Optimization
SP	General End Service Provider
VoIP	Voice over IP
VSP	Video Service Provider
XMPP	eXtensible Messaging and Presense Protocol

Chapter 1

Introduction

Even as the current Internet enables a range of services and distributed applications that grow ever broader and more vareigated, several limitations of its architecture have become apparent as billions of humans and devices are connected through it. This has caused a resurgence in interest in architectural research to address evolutionary paths for the Internet in recent years. National research directive and funding organizations have created programs encouraging such research in various contexts, including the Future Internet Architecture (FIA) program of the US National Science Foundation [49].

One key challenge is the discrepancy between the mechanisms by which technology is deployed in the Internet and the business models surrounding these processes. ChoiceNet [189, 27], one of the projects being conducted under the FIA umbrella, and that our research group is contributing to, attempts to address this challenge. Briefly, the ChoiceNet project proposes the introduction of architectural entities into the current architecture of the Internet to enable an "economy plane" that allows the presentation of competing offerings for various networking services, the formation of contracts for the various services that make up the entirety of a user's network needs, and tracking the performance of each provider in meeting their parts of the contracts. In Chapter 3, we provide a more detailed overview of ChoiceNet.

Contracts are meaningless without the ability to at least verify, and possibly enforce, compliance. From the service provider's point of view, such verification and enforcement often comes down to access or admission control of some sort. However, from individual or small customers' point of view, there is little or no granularity in the ability of obtain information about whether the contracted service was obtained, and if not, which one of the many service providers involved in providing the customer's experience are to blame. In some cases, a customer might be able to directly measure quantities of interest that provides the answers, but in many cases, the service experienced by the customer is composed out of several component services provided by distinct providers and such a provider's monthly average measurement statistics [178, 179] are of little to no value to diagnose the source of such problem. In the ChoiceNet view, the customer makes separate contracts with each such provider, and needs monitoring information for each service – some of which can only be obtained at points of presence inside the network cloud, where the customer does not have presence. This points to the need of a specific class of service providers: *independent third-party verifiers.* Such verifiers would have distributed infrastructure and points of presence, but instead of selling transport service, would sell monitoring service to customers.

In this work, we investigate a solution to this fundamental missing piece of the Internet, i.e., the measurement and verification capability of the network services offered in the Internet, that indirectly pushes more responsibility to the network providers to fulfill their requirements for high quality services.

In order to better understand the overall work in this thesis, we give a literature overview of the research relevant to our work in Chapter 2, which includes the existing and ongoing future Internet architecture research, the existing measurement support from routers, measurement frameworks and systems and an overview of the measurement tools developed by the research community. Chapter 3 first discusses the architectural design principles of the ChoiceNet and then proposes a verification service architecture with a detailed description of each role, interaction and interface as well as a discussion on measurement specification and the potential business model.

Having described the design in detail, Chapter 4 gives the results of a series of tests we have conducted to investigate the capabilities and benefits of using a measurement and verification service in wired and wireless domains. In Section 4.3, we articulate the simplest possible scenario to perform a verification service that is useful and meaningful, and we contrast how much simpler such a service is to realize when we assume standardized verification plane roles and interactions. As another facet of this, we show the power of the separation of roles we propose, by showing how a significantly harder verification task can be achieved by the same basic building blocks as for the simpler verification task, by providing tests in NS-3 simulator. Following the promising results, in Section 4.4

we show the results of an an actual prototype we deployed on a real system, the GENI meso-scale testbed. Section 4.5 details our contribution to the implementation details of a minimum yet complete ChoiceNet prototype with verification service.

Next, in Chapter 5, we investigate and validate the benefits of having informed choices based on a practical and useful domain; energy efficiency as a case study. In this study, we give the simulation results and analysis of an on-demand optical path provisioning service, where the users can either choose the inexpensive and energy-efficient path or the minimum delay but expensive path on optical networks. Finally, Chapter 6 discusses an incremental and scalable model for deploying this architecture as a real world system.

Chapter 2

Related Work

In order to better appreciate our work explained in the following chapters, several existing technologies and methods need to be explained first, and in this chapter, we give an overview of such literature. Figure 2.1 gives a high-level view of the areas related to our work; a general discussion on the future Internet research, followed by existing measurement support from routers and then the literature overview of the measurement systems. We finally briefly talk about the techniques for efficient packet capturing and well-known measurement techniques.

2.1 Future Internet Architecture Research

The Internet has started as an experimental network 40 years ago but has quickly grown into a commercial network that connects businesses and people. This success story is due to the Internet's well-prioritized design goals that kept the network simple and has led to many innovative applications at the edges, such as email, WWW, file sharing, VoIP etc. However, in the last decade, the Internet's architecture is challenged in many ways and this has led the research community to investigate new architectural designs that can better suite the current and future demands of the Internet, called "Future Internet Architecture" research. We divide the overview of this research into three categories; 1) Earlier clean-slate designs described in Section 2.1.1, 2) Current Future Internet Architectures described in Section 2.1.2, and 3) Cross-Layer Design overview described in Section 2.1.3. But, in order to understand the motivation behind future Internet Architectures, we first briefly explain the underlying principles of current Internet along with the new

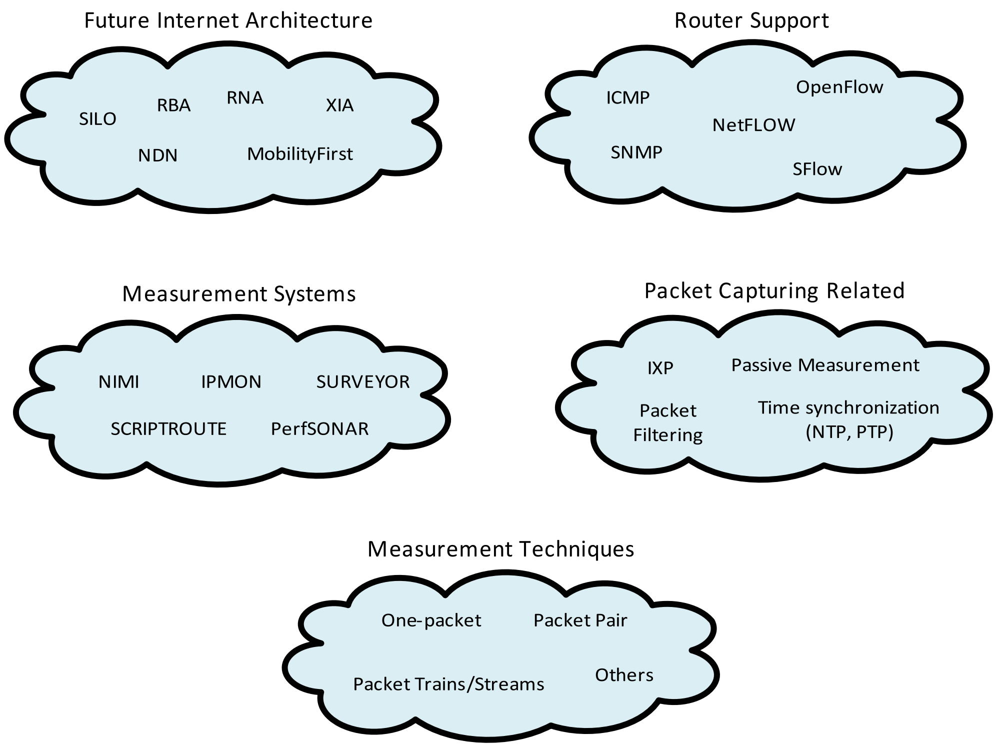


Figure 2.1: A high-level view of the relevant literature areas and examples

challenges it is facing today. We then give an overview of the work in the literature to solve these problems.

Current Internet Architecture: Started as an experimental network for researchers four decades ago, Internet has now become a world-wide commercial product and information infrastructure [98] that has been constantly changing the way we communicate, learn, entertain, make business and live in general. The Internet's well-defined goals [34]; "connecting existing networks", "survivability against network failures", "support for different services and physical networks", "distributed management", "cost effectiveness", "low barrier for host attachment" and "allowing accountability" were defined in the order of importance respectively and to meet these goals, several design principles [47]; "layering" to reduce complexity and flexibility, "packet switching" for scalability and cost-effectiveness, "inter-domain routing" (such as BGP) for distributed management, "configuration protocols" (such as DHCP) for low host attachment barrier but most importantly the end-to-end principle [155] to keep the network simple and end-nodes intelligent were introduced. The end-to-end principle deserves special attention because it made the network unchanged for new applications which opened up the path for innovations (such as WWW) and made Internet so successful.

Challenges: The prioritized goals and design principles that meet these goals were the key to Internet success for four decades. Especially in the last decade, however, the changing nature of users has challenged the Internet in many ways [17, 47, 35, 169, 82] such as security problems (Denial of Service attacks, IP address spoofing, spams, worms), mobility issues (coupling of host identity and location in IP addresses and inefficiency of mobile IP), quality of service (difficulties in providing certain performance guarantees in a stateless network), the lack of support for cross-layer interactions which reduces performance significantly, and also high management costs (instead of a self-configuring, optimizing networks). These challenges were not considered in the design goals and adding these functionalities usually led to the violations of the design principles such as half-layer solutions (such as MPLS, IPSec) [183] and cross-layer interactions [169] eroding layering, and middle-boxes (NATs, firewalls, web caches) deteriorating the end-to-end principle. While such violations of design principles with quick fixes seem reasonable, it has significantly reduced the flexibility of the architecture causing ossification of the current "the Internet Protocol Stack" [83], or simply TCP/IP stack, and consequently this problem led the research community to reconsider the underlying principles and think about clean-slate network architectures [54, 17] recently.

2.1.1 Clean Slate proposals

Due the challenges described above, NSF FIND (Future INternet Design) research program [54] asked for new Internet architecture proposals. An earlier work, X-kernel architecture [73], though not a clean-slate design itself, has commonly been referenced by clean-slate architectures [172, 89] due to its focus on modular messaging between protocols with service interfaces and it forms the basis for new ideas. RBA (Role-Based Architecture) [21] proposes a novel non-layered, heap-based approach with fine-grained services called "roles" where packet headers include role-headers with no hierarchy, as opposed to the traditional stack-based approach. This approach gives high flexibility but its additional cost must be investigated carefully. SOA (Service-oriented Architecture) [89] is an object oriented protocol model enables dynamic protocol instantiation by utilizing predefined and atomic functional blocks (FB), such as CRC (Checksum Redundancy Code) with an emphasis on defining relationships between protocols. RNA (Recursive Network Architecture) [172, 171] introduces an architecture based on a single, flexible meta-protocol as the building block, where one building block is reused on top of another to create the desired functionality and to avoid repeating the same functionality. GINA (Generalized Inter-Networking Architecture) [82] focuses on the general Internet architecture by introducing "objects" as addressable units that belong to "realms" as administrative domains with several servers to support better security and mobility, whereas "zones" represent the address hierarchy. In addition, SILO (Service Integrated Control and Optimization) [13, 183, 177, 150] is clean-slate architecture developed by our research group, explained in detail in Appendix A.1.1.

2.1.2 Recent on going work

The next effort to make better future network architectures is the Future Internet Architecture (FIA) program of the US National Science Foundation [49]. There are five ongoing future Internet projects [10] supported by FIA; Named Data Networking (NDN) [120] has an emphasis on the content-based networking instead of the traditional server-based networking, whereas Mobility First [114] focuses on fast ID to address translation, self-certifying public key addresses and routing for mobile nodes. On the other hand, Nebula (latin word for cloud) [121, 122] is a cloud-based architecture where all services are provided from the nearest data centers, whereas XIA (eXpressive Internet Architecture) [190] focuses on the extensibility by naming everything (content, host, service) with XIDs (eXpressive IDs) which the network can find the closest copy upon request. Finally, our ChoiceNet project (discussed in Section 3.1) aims to improve the capabilities of the Internet by introducing "choice" for network services and it is orthogonal to the projects mentioned above.

While the research on future network architectures continues, there are concerns about the dynamics shaping the future of the Internet [36] and usefulness of clean-slate architectures [147], such as the economic infeasibility of changing all the networks deployed today and also replacing an existing system (tested and debugged for 30 years) with a brand new system. In order to address these concerns, realistic and large-scale testbeds [54, 63, 53, 86] are necessary to test advantages and problems of new approaches, such as GENI, discussed in Appendix B.1. In the next subsection, we describe basic cross-layering concepts.

2.1.3 Cross-Layer Design Overview

With the emerging technologies such as wireless networks and mobile devices, the application and network performance is now more dependent on new constraints such as battery lifetime or variations in the wireless networks, and cross-layer paradigm addresses these new challenges by interaction across layer boundaries to improve the performance [145]. A few examples to give insight are; 1) the wireless channel information such as SNR (Signal to Noise Ratio) can be utilized to adjust the sender power level for efficiency in a video multicast streaming scenario [182], 2) the interaction between link layer and application layer can assign different priorities to different application data to improve Quality of Service, or 3) interaction between networking layer and transport layer can be utilized to avoid TCP throughput reduction in a hand-off case. While useful, ad-hoc cross-layer interactions are typically achieved by making assumptions about the inner workings of a different layer making the architecture even more ossified. To address this issue, "cross-layer architectures" [166] have been proposed in order to provide systematic ways for such interactions. Shared database approach [4, 188] is particularly attractive because it enables optimization programs to query and interact with all layers through standard interfaces without exposing internal layer details. We describe this phenomenon in SILO in Appendix A.1.1, which has built-in support for cross-layer interaction and optimization.

2.2 Measurement Realization Technologies

This section covers a broad overview of the technologies available today to make measurements in practice. The large number of technologies and concepts that relate to this, show that ChoiceNet Verification Service Architecture (discussed in Section 3.2) would have a rich ecosystem. But at the same time, it further underscores the need for having an underlying architecture of a few standard roles, and a common set of interactions between them. Due to the lack of such an architecture and the key insight that different roles in the architecture can and should be played by different business entities, none of these technologies by themselves have been able to introduce a commonly used framework in the Internet. The strength of our unified architecture is the potential to enable both existing and also new technologies to coexist in the same service ecosystem and motivate business entities to provide these services, unlike what happens in the current Internet today.

We first start with the protocols router vendors support, followed by a brief survey on measurement systems in the literature and passive measurement technologies. We finally discuss the existing measurement tools and methods.

2.2.1 Router Support for Network Measurement

While measurement services do not necessarily depend on router support, such popular and widely-deployed technologies offered by router vendors can add value to the services offered by providers.

ICMP (Internet Control Message Protocol) [138, 92] is the fundamental protocol to troubleshoot connectivity issues on IP networks such as network or host unreachability. Upon diagnosing such problems, routers send an ICMP error message back to the source host. Additional error messages include time-exceed message due to the expiration of time-to-live (TTL) value (practically used as hop count field) of IP packets [139] and parameter problem messages. In order to find out connectivity issues with a node in the network, ICMP request and reply messages are used. Moreover, ICMP timestamp and reply messages are used for time information. Additionally, Route Record IP Option makes routers along the round trip path add their the outgoing IP addresses to the IP Options field, up to 9 IP addresses [167] due to space limitation at the IP header. Such ICMP messages, IP Options or IP TTL fields have been extensively utilized by the research community to extract measurement data from the network due to its simplicity and availability, as discussed in Section 2.2.4.

While every router must implement ICMP capability and process IP Options [139], in practice many network administrators choose to rate limit [12, 158], discard [95, 102] or set low priority [109] for these messages at routers for security and performance reasons. Thus, utilizing such messages is simple, but its accuracy is questionable.

SNMP (Simple Network Management Protocol) [24, 92] is the fundamental and widely-deployed network management protocol developed in order to handle the network management and monitoring complexity. SNMP is based on a manager periodically polling information from network agents typically located at routers and switches. This way, the node and interface level information, such as number of packets received or link utilization can be gathered easily from all network nodes, without putting heavy computational load on the nodes or adding extra load to the network.

While the aggregated information SNMP provides is useful for capacity planning, it helps little to characterize the traffic patterns, which is important to understand whether quality of service requirements are met to support businesses. Therefore, we believe SNMP can only be used as a sanity check for most measurements. Furthermore, SNMP information is typically accessed only by the network administrators and making this information available in real-time may not be always possible.

While ICMP and SNMP are helpful and widely-deployed, the quality of service and security requirements need more detailed information about the traffic passing through the networks, so new protocols are offered by vendors as we discuss below.

SFlow [77] enables extraction of valuable traffic pattern information at very high speed networks, such as IXPs. SFlow achieves this scalability by sampling packets received at network nodes and sending the header of sampled packets to a central entity. SFlow Agents are located on routers and switches, and these agents send this information to a central SFlow collector by UDP. While low sampling (such as 1 in 16K packets) decreases the accuracy of results, SFlow is proved to be useful even in large IXPs [2, 84] to catch basic patterns such as how much of the traffic is based on HTTP or IPv6. Moreover, many service providers are exchanging large amounts of traffic in such IXPs, and the ratio of ingress to egress traffic between two providers can be retrieved with this sampling technology. Although SFlow is highly scalable and is a good candidate for understanding general traffic patterns, we believe it is not adequate for flow-specific information due to its low sampling rates.

NetFlow [31] is Cisco's proprietary technology providing flow-specific information useful for various purposes [29] such as analyzing network traffic of new applications, user profiling, security analysis and validation of QoS parameters. NetFlow is commonly deployed in routers and switches today, where a flow is defined as a unidirectional stream of packets with the same source and destination IP, transport protocol, source and destination ports and device interface. It can detect the end of a flow by TCP FIN or RST bits within a packet. NetFlow can export the number of packets and bytes of the flows from the network nodes to a "NetFlow Collector". The packet format used to convey the flow information consists of either fixed fields (version 5) [28] or template based fields (version 9) [32] to enable flexible parameter negotiation (such as sampling interval and algorithm), memory savings and extensibility. Unlike SNMP polling, NetFlow agents push information to the NetFlow collector periodically to save network bandwidth and uses [31] only 1-5% of the traffic for exporting NetFlow data. NetFlow typically uses UDP as the transport protocol but also supports SCTP.

An earlier NetFlow performance analysis [30] showed that 65000 different IP flows add only reasonable CPU load on most of the routers and sampling greatly reduced the CPU load in routers for both version 5 and 9. The same study also claims that 1:100 sampling (1 out of every 100 IP packet switched) identifies approximately 80% of the flows.

As the logical successor of NetFlow version 9, IPFIX (IP Flow Information Export) [33] is an IETF protocol emphasizing on a standard representation of flow data to be transmitted from the flow exporters to the collector, with more flexibility and extensibility in mind. IPFIX also supports security mechanisms such as TLS, DTLS, X.509 as well as several encodings [141] including XML. Furthermore, combinations of sampling and filtering sequence [153] can be defined as rules to select incoming packets for measurement, where sampling can be specified for packets out of a particular flow or out of all flows on a link [142]. IPFIX provides a standard for the protocol only, leaving the implementation details [173] to the vendors.

Unlike SFlow, NetFlow can extract statistical information also without sampling, i.e. from all packets of a flow. While this seems attractive, it puts significantly high load [30] on the CPUs of switches for high speed networks so it may not be sufficient for IXPs (discussed in Section 2.2.3).

OpenFlow [110, 130] is the recently emerged software defined networking technology, which enables better control and management of the flows going through the network as well as the collection of per-flow measurement data. OpenFlow achieves this functionality by separating the data plane and control plane of the commercial Ethernet switches and routers, without vendors exposing their code. The control plane functionality is performed by the remote entity called the “OpenFlow Controller”, which makes decisions and sets rules about the network flows, such as forwarding a particular incoming flow traffic to a particular switch interface. These rules are then performed by the OpenFlow-enabled switches. Such centralized decision making not only simplies the traffic engineering but also better utilizes the network resources and reduces the network costs. In addition, OpenFlow switches have a limited support for quality of service with multiple queues.

From the measurement perspective, OpenFlow switches have per-flow counters [127] to provide measurement data, such as the number of packets or bytes received for each flow. However, the maximum number of flows supported by switches are typically less than 100000 [128], and therefore administrators usually aggregate many five-tuple flows into a single flow represented by a network prefix, VLAN or MPLS tag, which makes the analysis of individual five-tuple flows difficult. On the other hand, the main power of OpenFlow is to enable on-demand path-setup capability. Because OpenFlow controller has a complete view of the network flows, it can use this information to first determine the path for a new flow with specific quality of service parameters and it can then setup such a path for the new flow as a set of rules across the switches.

A technology similar to OpenFlow used in data centers to control virtual machine flows running on host server machines is OpenVswitch [129]. In addition, following the OpenFlow technology, FlowVisor [161] enables slicing of the network resources among multiple OpenFlow controllers so that each controller controls its own isolated portion of the network, which may create a more competitive environment for network services. Complemented by a verification service, we believe that OpenFlow or similar technologies have the potential to provide a better value for service providers and customers.

2.2.2 Measurement Systems

In this section we give a broad overview of measurement systems including measurement infrastructures, architectures, frameworks or APIs, which utilize existing measurement tools and orchestrate nodes to make measurements.

NIMI (National Internet Measurement Infrastructure) [135] is one of the earlier measurement infrastructures deploying end nodes on the network and making these nodes publicly available for active measurement with popular measurement tools (discussed in Section 2.2.4). Authors also provide guidelines for deployment and management of nodes, including the necessity for a high level of abstraction such as a service layer, to better utilize measurement points (MP), which otherwise gets too complicated and too hard to orchestrate. Besides, authors kindly provide their invaluable experiences [133] by emphasizing on the usage of portable and safe programming languages as well as insights about the software maintenance and management problems.

OCXMON is responsible for the passive measurement part of NLANR (The National Laboratory for Applied Network Research) [108], an organisation which develops the Network Analysis Infrastructure (NAI) for measurement and analysis, and also includes the vBNS and Abilene high performance research networks for HPC (high performance connection) community. OCXMON project makes passive measurements by tapping into optical links through optical splitters technology. Also OC3MON [61] (earlier version of OCXMON) demonstrated a showcase that passive measurement on an OC3 (155Mbps) link with commodity computers was possible. It is important to note that this demonstration [61] captured all of the 80 million packets received within the time interval and provided excellent traffic information.

AMP (Active Monitoring Project) [109] is the active measurement part of NLANR, which deploys end nodes on the network with custom measurement software. Considering the remote management of all nodes (except booting), an interesting point the authors make is that having higher quality hardware can sometimes reduce maintenance cost. Also, the deployed end nodes do not have GPS for the sake of easy deployment and the authors claim that the system is still valuable, since several TCP-based applications such as Telnet, Web or FTP typically care about RTT (round-trip delay) measurements. In addition, AMP has a "notification" feature which sends notifications to interested parties in case of an abnormal observation in measurements, such as a spike in RTT. Also disk storage must be carefully considered for continous measurements.

IPMON (IP monitoring system) [59, 60] is a distributed monitoring system in order to capture the traffic patterns and also merge the data measured at different points of network. Authors point out the drawbacks of active measurements; 1) injecting probe packets can bias the measurements, 2) active measurements do not archive other metrics and 3) it is hard to specify a path for such probe packets. IPMON system is deployed on Sprint PoPs (Points of Presence) with packet capturing on OC-48 links (2.5Gbps). GPS is used on all nodes, creating 1.1 TB data daily with 64 byte records of 44 byte packet headers. While IPMON is not a service-based system, it still has the three phases: "measurement data production", "sending it to a central database" and "analysis". IPMON has a focus on general traffic patterns such as the percentage of TCP packets or the percentage of fragmented IP packets.

SURVEYOR [66, 88] is an active measurement system using commodity computers with GPS support. This is especially valuable for one-way delay because even symmetric paths may give different results due to the traffic load on different directions, especially on Trans-Atlantic or Trans-Pasific paths. The authors also modified the end point network drivers so that the incoming or outgoing packets are timestamped in kernel with accuracy in 10s microseconds range. With this mechanism, the authors were able to detect a forward path with 7ms and its reverse path with 75ms one-way delay. This clearly shows the importance of one-way delay measurements and the need for GPS. Surveyor also follows the three phases discussed in IPMON above.

SCRIPTROUTE [165] is an active measurement system offering more flexibility than NIMI at vantage points. While NIMI required credentials to use the deployed nodes, Scriptroute wants to minimize the user-barrier by allowing unauthenticated access to the measurement nodes. The system provides security by requiring user applications to be written in Ruby (interpreted, safe and portable language) and be run in a sandbox in order to limit the traffic it generates, the time the experiment runs and also to avoid any local storage. Although this seems like a drawback, the authors claim that these limitations are reasonable for most of the measurement tools (discussed in Section 2.2.4). We believe this is a valuable earlier attempt for active measurement points.

TULIP [105] gives a measurement architecture guideline starting from the ideal world where all incoming packet information is logged at routers, to a practical sampling approach by leveraging existing tools such as ICMP Timestamp for timing and IP Identification field for distinguishing packets. However, the limitations of ICMP Timestamp are discussed in Section 2.2.1 and IP Identification field is sometimes insufficient to uniquely identify packets, based on the experience [60, 186] due to its high implementation dependence and 16-bit field. Thus, such mechanisms are not reliable enough for accurate measurement results.

DIPZOOM [143] proposes a service-based active measurement system with a marketplace, which works as a "match-maker" for the "participants" in order to make active measurements in a peer-to-peer fashion, where participants set the prices for their "services" and clients request these services. Using DipZoom API, clients first retrieve the list of available nodes to communicate and then start experiments on those nodes with standard tools such as ping (discussed in Section 2.2.4) or wget [174]. Dipzoom utilizes UDDI format, WDSL Web service standards and SOAP messages for capability negotiation, but authors do not provide any details about the specifics of the messages thus we assume it has a limited negotiation scope, i.e. only for the parameters of the well-known measurement tools instead of a comprehensive description language.

We have also noticed later that, our work has a similarity with Dipzoom in the idea of "measurement credentials", that is, after the measurement request and payment stages, a credential is provided for each measurement node (or measurement host in Dipzoom terminology) for further communication. Furthermore, the authors point out the possibility that measurement points may indeed fake the results or advertise more capabilities than they actually have, in order to make more money. While introducing the concepts of marketplace and payment system for the first time, Dipzoom has no notion of service composition, analysis or an independent verifier making measurements in the middle of the network.

REM (Reactive Measurement Framework) [6] is a software framework running on the client's computer, based on the idea that measurements are a process rather than an event, and therefore several measurements can be executed one after another in an automated way in order to find out the connectivity issues. For example, a webpage load problem can be diagnosed better, if a traceroute (discussed in Section 2.2.4) measurement is accompanied afterwards, rather than manually running such measurements. The logic of the measurement selection is provided by a configuration file, but it would be much better if the authors had also provided an example of such a REM instance. We believe a software similar to REM can be used to request measurements from our architecture.

UANM (Unified Architecture for Network Measurement) [1] is a measurement architecture, where each measurement node has a "knowledge base", a "daemon" and a “decision engine”. A knowledge base is a configuration file which includes a) the available “measurement plugin”s with their characteristics, b) a daemon list to communicate with other nodes about their capabilities and network conditions. After querying the knowledge base, a daemon accepts requests, negotiates capabilities and then loads the necessary “measurement plugins”, which are loadable measurement tools doing the actual measurement task. A decision engine determines the feasibility of the request and schedules the plugins in order to handle concurrent requests typically in a FCFS (first come first service) fashion so that the interference between measurements is reduced.

However, the authors have not mentioned any discovery mechanism, except the default information knowledge base, which may cause scalability and maintenance issues. Also, there is no notion of credentials or even analysis for measurements. Additionally, the authors showed that a policy such as FCFS is helpful to avoid interference at a single end node, but we believe this still has a limited benefit because several nodes may be doing similar measurements through the same tight link causing still high cross-interference. This problem is in fact difficult to avoid when a large number of concurrent active measurements are performed in the network. Furthermore, it is not clear how new measurement tools may be loaded into these nodes. We believe these are important questions to be addressed, in order to create an extensible, easy-to-use, secure and scalable architecture.

OML (Orbit Measurement Library) [111], which was initially a component of OMF (discussed in Section 4.2), can now be used independently as a measurement library. With repeatibility and a common format in mind, OML works by calling the OML API at the desired source code line of the experimenter’s application, which actually adds measurement point capability to the application and this API then connects to an OML Server using TCP to publish measurement results during the experiment. Also, the authors show that OML does not have any noticable overhead for CPU and memory.

ICING [119] Although not a measurement architecture, ICING proposes a “Path Verification Mechanism (PVM)” which enforces that path policies are indeed followed. We believe this is a relevant and important technology to our architecture because it can enable users to have some control over the path policies for their packets, as long as the network providers give consent. In order to achieve such a capability, a sender puts a “proof of consent (PoC)” and a “proof of provenance (PoP)” information into each packet. A PoC is a cryptographic token guaranteeing that the consent is given by the provider’s consent server, whereas PoP fields are modified by ICING nodes on the path, to prove that the packet followed the given path. Several cryptographic mechanisms are used in order to make ICING both secure and also efficient, and the authors claim that its implementation performance and cost is reasonable. In addition, one of the advantages of ICING is that it does not require a central authority or PKI to distribute public keys.

S^3 Monitor (Scalable Sensing Service) [16] is a measurement service that enables easy measurement setup and measurement data collection for GENI experiments (discussed in Appendix B.1). S^3 utilizes standard measurement tools such as ping or traceroute (discussed in Section 2.2.4), run by measurement points (or “sensor pods” in S^3 terminology) on the reserved nodes of a GENI slice. Sensor pods collect the measurement data during the experiment and then send it to a central database, which the experimenter can later access and view the results from a web-browser. An elegant feature of S^3 is to utilize the RSpec (resource specification file for a slice request) to deploy the measurement nodes easily and dynamically. With extensibility and portability in mind, S^3 specifies the sensor (such as ping) capability description in detail and it has the sensor pod code written in Python language for better portability across heterogenous GENI platforms. The primary difference between S^3 and our architecture is that S^3 measurements are specific to the experimenter’s flows at the end-nodes but it is not concerned with the measurements in the middle of the network.

PerfSONAR (Performance focused Service Oriented Network monitoring ARchitecture) [68, 156] is a popular measurement architecture which provides network measurements across multiple domains and also enables easy retrieval of measurement results for network managers and even for end users. This capability is especially important for troubleshooting and ensuring that QoS (quality of service) parameters are satisfied. To the best of our knowledge, the work in the literature that attempts a task closest to architecting a complete measurement ecosystem is the PerfSONAR project due to its service oriented (SOA) and flexible architecture.

Its SOA based approach improves robustness and flexibility by independent entities each providing different services.

PerfSONAR architecture consists of three layers; (i) the "measurement layer", where measurement points (MP) perform the active or passive measurements for various metrics (such as one-way delay, jitter, link utilization, loss), (ii) the "service layer", which provides "services" to users such as discovery, security, measurement services and which is also the glue between administrative domains, (iii) the "user interface layer", where the end user tools visualize the measurement data [85]. Because everything is a service in PerfSONAR, we now describe PerfSONAR by the services it consists of;

Lookup service (LS) is the entity, which other services register and then clients query LS in order to discover such services along with the queries clients can make.

Authentication service (AS) gives the credentials the clients need for authentication across multiple administrative domains.

Measurement Point Service (MPS) makes active or passive measurements based on the requests, using wrapped tools such as PingER for ping, BWCTL for Iperf [58] or other protocols such as SNMP, NetFlow. MPS then provides the measurement data (MD) as a service.

Measurement Archive Service (MAS) is the storage unit with a database optimized for MD, such as a round-robin database (RRD).

Transformation Service (TS) is the MD post-processing or analysis unit, responsible for aggregation, correlation and filtering raw MD.

Topology Service (ToS) is responsible for determining the proximity and topology of MPs to have the network topology on map.

Information Service (IS) is responsible for both Lookup Service (LS) and Topology services (TS), after they conceptually merged into a single IS [20].

Resource Protector Service (RPS) ensures that limited resources such as bandwidth, is not affected during a measurement by restricting access to these resources and accepting requests based on credentials and resource availability.

The interactions between PerfSONAR entities are based on Web Services for extensibility, and also Apache Axis implementation with SOAP messages is chosen in order to simplify client implementation. That is, creating and parsing XML messages are easier for clients, compared to other distributed SOA implementations. New metrics can easily be added to PerfSONAR as we have shown in Section 4.1, by extending a perfSONAR user interface tool, called PerfsonarUI [85] and in addition, by adding support for real-time optical measurement data retrieval. While a basic example for metric composition for PerfSONAR has been suggested in [99], to the best of our knowledge, a comprehensive study about metric composition is not suggested later for PerfSONAR.

In our envisioned architecture, perfSONAR entities and mechanisms can participate seamlessly as MSPs (measurement service providers).

Argos [149] is a distributed measurement framework for wireless sensor networks sniffing 802.11 traffic at outdoor urban areas within 10km^2. Argos faces unique challenges such as finding the channel to sniff the right packets, the signal quality variation at different locations and encrypted traffic. Argos addresses these issues by (i) filtering packets with libpcap [176] based on the user requests, (ii) coordinated hopping to a particular channel in order to sniff the same flow at neighbor nodes and then merge these traces for better measurement data, and (iii) guessing user locations by 802.11 probe messages even in encrypted traffic. In our architecture, Argos can be an MSP and it is even possible that the users may give a temporary password such an MSP so that it can decrypt the sniffed traffic.

Evaluation: Though the architectures surveyed above have certain useful capabilities, they do not represent a realistic business model, that is beneficial for both customers and providers. Thus, there is little motivation for a commercial support to widely deploy them. As discussed in Section 3.1, it is essential for any new architecture to provide such motivation to be successful. We also believe that, as customers have more choices, they will demand more customer-oriented analysis, in order to choose "better" providers to maximize their quality of experience. In Section 3.1.2, we discussed our design and interactions to maximize such a motivation, and now we continue with the popular passive measurement techniques used in practice.

2.2.3 Passive Measurement in Practice

There are several different aspects of passive measurement and in this section, we first describe popular packet capturing technologies. We then give a short discussion on the Internet Exchange Points (IXP), followed by popular time synchronization techniques in practice.

Efficient packet capturing techniques

We believe it is important to emphasize here that we are not interested in high-speed packet forwarding, but only capturing packets passively and filtering them efficiently. Also, we are not necessarily interested in the packet content, i.e., packet headers (with the packet length field) are usually adequate for general measurements (delay, throughput, packet loss etc). Below we give some of the popular packet capturing mechanisms in practice.

Using optical splitters, the OCXMON [108] project has shown that passive monitoring at high-speed optical links is indeed possible, even a decade ago. Today, DAG cards [39] offer packet capturing at 10GigE or SONET network interfaces [60] and can be utilized. Similarly wire-tapping [37] has been used for passive measurement. Additionally, routers and switches typically have mirrored ports to provide a duplicate of packets going through the network, however this adds additional load on these devices.

A study [162] showed that modifying a 10GigE NIC and its device driver in order to capture all packets at commodity computers is possible by sending only the packet headers with timestamps to the computer memory. Moreover, using the same NIC and driver, the authors also showed an efficient traffic generation method, which replicates the exact "captured" traffic by sending the same size but "dummy" packets. This is a valuable work proving that passive measurement is possible at 10Gbps with commodity computers.

Another work [159] utilizes existing switch protocols to distribute a 10Gbps traffic into 10 separate 1Gbps interfaces so that 10 commodity computers can process them efficiently. The authors claim that the timestamp issues of packets belonging to the same flow but arriving at different computers can be addressed by sending the same flow traffic to the same interface.

In high speed networks, packet filtering is a critical step for the performance of the measurement point, and that is why, efficient usage of system resources should be ensured. BSD Filter [107] is a popular technology to filter and timestamp incoming packets at kernel-level while adding minimum load to the system by minimizing the memory copying and saving CPU cycles. In addition, CoralReef [91] is a generalized libpcap [176] and its API provides a unified interface for getting measurement information. It is also important to note that once a packet is timestamped correctly, the time it later takes to do any additional processing on the packet does not change the accuracy, so measurement mechanisms can be run in user space. A recent work on filtering, called PFQ [19], shows that the multi-core nature of commodity computers can increase the packet capturing rate with multiple threads, compared to BSD Filter, even on 10GigE cards.

The performance comparison for operating systems [144] show that RTAI [152] (a Linux extension for better real-time performance) is a good candidate to be run on the measurement points, with its low latency and response times than regular Linux.

Internet Exchange Points

We envision that Internet Exchange Points (IXP) are good candidates for deploying measurement points and that is why, it is important to understand IXPs. There are around 350 IXPs worldwide [3] and significant volumes of Internet traffic is subject to IXP effects. For example, a large European IXP, Amsterdam IXP carries 10 petabytes of traffic in a day [2], and there are around 400 members connected to this IXP. Access routers of these members are connected to each other by the IXP layer 2 switching fabric and if peering is desired, BGP sessions are establied between such members. In addition, recent IXP traffic patterns are interesting such that only 3% of AS'es consume 30% of the total traffic, and also 30% of AS'es carry 90% of the whole traffic. Furthermore, most of the incoming and outgoing traffic between peers are highly asymmetric, according to the SFlow sampled data. We believe such studies are valueable at selecting the links to deploy the measurement points, if measuring all links at the same time is not economically feasible. The study also shows that more than half of the traffic is web-based, as expected.

Another study also [59] gives insights about a large network provider's backbone and point of presence structuring.

Time Syncronization Technologies

Time synchronization and accuracy are essential for accurate measurements especially for one-way delay, and in order to provide the synchronization among entities, a highly-accurate (less than millisecond) clock synchronization technology is necessary. In this subsection, we discuss the three popular clock synchronization technologies; NTP, GPS and IEEE 1588.

Network Time Protocol (NTP) [112] is a common protocol for time synchronization and distribution in wide area networks. It has been extensively deployed on commodity computers because it only requires software support. NTP RFC [112] defines several clock terms that we find useful. For example "clock stability" refers to how well the clock maintains a constant frequency, whereas "accuracy" refers to the ability to keep up its frequency and time with the national standards. Another term "precision" refers to how precisely these values can be maintained within a timekeeping system. Also, "offset" of two clocks is the time difference, whereas the "skew" is the frequency difference (first derivation of offset). Finally, "drift" refers to the variation of skew, i.e., the second derivative of offset.

NTP servers work in hierarchy, where the most accurate and precise time servers are called "stratum 1" providing synchronization to less accurate servers called "stratum 2" and so on. Stratum 1 servers are the primary time sources, typically utilizing a reference clock directly traceable to UTC such as GPS [113].

An NTP message exchange between an NTP client and server provides four timestamp values to the client; T_1, T_2, T_3, T_4 for client transmission time, server reception time, server reply transmission time and client reception time respectively. Using these values from various servers, clients can find a set of clock offsets (the difference in time with respect to reference clock) as $\theta = ((T_2 - T_1) + (T_3 - T_4))/2$, round trip delays as $\delta = ((T_4 - T_1) - (T_3 - T_2))$, and dispersions as $\epsilon = \rho_R + \rho + \Phi(T_4 - T_1)$, where ρ_R, ρ, Φ represent server precision, system precision and tolerance constant respectively. Then, NTP client utilizes several algorithms [113] to find the time closest to the actual accurate time. The first of such algorithms is the selection algorithm which rules out the "falsetickers" (incorrect time values) based on Marzullo's algorithm [106] by finding the intersection interval from several offset and root dispersion values. Having ruled out the incorrect values, the clustering algorithm then filters the statistical outliers to find out the few best possible candidates by producing a "survivor list" and finally the combining algorithm weights each of the candidates according to the round-trip delay that its source has. Also, the "prefer" option enables user-defined preferences to have higher priority. Using these mechanisms, NTP finds the closest estimated value to the accurate time.

NTP timestamps are defined in unsigned fixed 64 bits, where first 32 bits represent seconds since 1 Jan 1990 and the last 32 bits represent the fraction part. NTP uses UDP as transport protocol. Because NTP messages usually go through a WAN (Wide Area Network), the accuracy is considerably affected by the network conditions. Therefore, while useful and inexpensive for general applications, NTP provides accuracy around 1-10 milliseconds range, which is too imprecise for millisecond-level accuracy, as suggested by several papers [85, 74]. Thus, in our architecture, NTP is only useful if the computers are close to accurate NTP servers.

Global Positioning System (GPS) is a widely popular time synchronization technology providing excellent accuracy. The one-way (or direct-access) method [100] mainly works by GPS satellites broadcasting time and position messages to GPS receivers, which calculate the distance through positioning and propagation delay, in order to correct their local time. Although the effect of the ionosphere, broadcast position accuracy or hardware delays can affect the calculation, the accuracy is still in 100ns range. While the costs of GPS receivers are reasonable, antenna and installation costs for buildings are still expensive. However, large IXPs typically have a GPS-based system installed.

IEEE 1588 Precision Time Protocol (PTP) is a standard protocol [74] providing precise time synchronization in local area networks, such as Ethernet LANs. With nodes that support hardware-based timestamping, unprecedented accuracy in sub-microsecond range can be achieved, whereas software based solutions also work with accuracy below 200 microseconds [48] in off-the-shelf PCs. PTP works by a master node first sending “sync” and “follow-up” multicast messages to the subnet. It is followed by slaves sending delay request messages to the master, which replies them with the reply message. These messages help slaves calculate both the round trip latency and software related (operating system) latency to calculate the offset. Intermediate devices such as switches can cause fluctuation in the results so these devices can act as “boundary clock”s, i.e., acting like a master to the slaves and acting as a slave to the grandmaster. The grandmaster is the main clock that all nodes in the network are synchronized to, and its clock is typically referenced to a stable and accurate time source, such as GPS. Therefore, a precise and accurate synchronization is achieved within the network in a cost-effective way even with software-based solutions. We believe this technology combined with GPS reference is a good deployment candidate for high-accuracy.

In the next section, we describe the existing measurement tools and methods in the literature.

2.2.4 Measurement Tools and Methods

A great deal of research work exists in passive and active measurement tools to estimate several metrics such as delay, link bandwidth, available bandwidth and packet loss. Understanding the concepts behind these tools are important and we envision that such tools can fit into our Verification Service Architecture as services. In this section, we give a broad overview of existing measurement tools and methods in the literature. We first begin with the basic mechanisms followed by measurement tools categorized by techniques. The categorization is based on the existing survey work [96, 140, 163, 187, 7] in the literature to easily understand the different approaches.

Basic mechanisms

In the current architecture, the only widely-used protocol enabling interaction between an end node and an intermediate network node, specifically a router, is ICMP (discussed in Section 2.2.1), and this is why, there are only a limited number of ways for measurement tools to get information from the intermediate nodes in order to generate measurement data. The most popular tool is "ping" [167], which uses the time between the dispatch of an ICMP echo request and the receipt of the ICMP echo reply to estimate the round-trip time (RTT) between the source and the destination. Ping can give accurate RTT estimates if routers (or end points) agree to reply timely, which is not always the case in practice as discussed in Section 2.2.1. If implemented correctly, ping is a simple and yet powerful tool since it only relies on the host time, without any synchronization requirement. Ping is also frequently used by network service providers [178, 179] to periodically check the RTT between core routers. Optionally, ping can use the "record route" IP option [167] to find out the router IP addresses along the RTT path between the source and the destination. This option makes routers add IP addresses of their outgoing interfaces, while the ping packet is traversing first from source to destination, and then the other way around. However, the IP header has space for only 9 IP addresses and that is usually not sufficient. Moreover, ping has an option for ICMP Timestamp [167] to estimate the delays for each router in addition to the router IP addresses on the path but this has even less space per hop, and there are also time format and synchronization issues. That is why, the timestamp option is not frequently used and protocols such as NTP have been developed (discussed in Section 2.2.3) instead.

Another popular tool is "traceroute" [79] which sets the IP TTL (Time To Live) field to a small value and relies on ICMP Time-exceed error messages sent from the routers on the path, when the packet IP TTL expires. Traceroute is widely used to discover the forward path and to measure RTT. However, the availability and accuracy of router responses are questionable as discussed in Section 2.2.1. Note that IP TTL messages are sent from router's IP address of the received interface [167] whereas IP Record Route option makes routers add the IP address of the outgoing interface, which may be helpful in finding more information about the router interfaces and the path packets travel.

One-packet Technique

One-packet technique refers to sending one packet for each measurement data. An earlier measurement tool using this technique is "pathchar" [80] which uses traceroute to estimate link characteristics by sending a series of probe packets from end points to each hop with various packet sizes and relies on the assumption that the minimum RTT received from a hop must experience the fixed link latency (propagation delay + packet size/bandwidth). Using this information, the authors also estimate the link bandwidths by taking the inverse of the slope of minimum RTTs for varying packet sizes. Another tool based on pathchar is "clink" [42, 41], which sends the probes adaptively in order to have a better estimation on the problematic links. Furthermore, by taking the difference between minimum RTT and all measured RTTs, it attempts to find a queueing delay distribution. However, pathchar and clink both estimate the link bandwidths by relying on the small time differences between results obtained from different packet sizes, assuming routers respond timely. For high-speed networks, such time differences will be even less accurate resulting in more errors in the estimates as observed in [95]. There are other drawbacks with these approaches; (i) sending around 3000 packets to perform a measurement adds significant load on the network if applied for many hosts [94], (ii) it takes time to converge, (iii) it assumes ICMP packets are treated equally, and (iv) highly inaccurate results are observed in multi-channel links such as OC-3 and DS1.

Packet Pair

Packet-pair approach [94] makes interpretations of the delay (one-way delay or RTT) difference between two successive packets sent back to back from the source. An earlier work [18] was using an "UDP echo tool" [157] between end-points to estimate end-to-end bottleneck bandwidth, RTT and loss behaviour by interpreting the correlation between the RTT of packet n and the RTT of packet $n + 1$. That is, in low load, the slope of increased RTT time difference between two successively sent packets gives the bottleneck bandwidth. The paper also attempts to estimate loss probabilities based on the assumption that losses happen in bursts as well. There are two drawbacks of this approach. First as [94] indicates this approach is based on the assumption that two successive packets will be queued back to back, which in reality might not necessarily hold and cause accuracy problems, and secondly the bottleneck link used in the paper was 128Kbps. However, in today's high-speed networks, the sender might not be able to timely send packets back to back within a very short time. For instance, if a sender sends two packets of 1250 bytes with 1ms separation, then the maximum bottleneck bandwidth the receiver can measure is 10Mbps regardless of the actual link bandwidth (say 100Mbps). But it is better than pathchar in terms of network load.

Due to the noise in the measurement results, several filtering techniques are proposed such as PBF (potential bandwidth filtering) [94], which assumes that the measured bandwidth vs. potential bandwidth graph must follow the $x = y$ line until a value $x = b$, and then follow the $x = b$ line, where b is the bottleneck bandwidth, and makes the filtering accordingly to find b.

Diagnosing the bottleneck bandwidth link is especially valueable for servers to better understand the problems with network performance. Nettimer [97, 95] combines the one-packet and packet-pair techniques to use the "tailgating approach" where a large packet is sent followed by a small packet in order to find out the queueing delays and the bottleneck bandwidth. While previous methods rely on ICMP replies in a timely and consistent manner, packet tailgating relies on a stream of packets got queued at particular points. It sends a large packet with TTL to expire and small packets followed by it. The authors also suggest "kernel density estimator algorithm" for filtering, which basically selects the dominant mode in the distribution. This method uses a considerable bandwidth though it is still less than pathchar.

Another popular packet-pair tool is Spruce [168] which is based on the "packet gap model", meaning that the additional gap between a pair of received packets is because of the cross-traffic and is the indicator of the available bandwidth. Spruce is a fairly accurate tool [163] considering the low number of packets it uses.

Packet Trains, Streams, Chirps

The "packet train" technique refers to sending a series of packets back to back from the source to the destination and interpret the network conditions using the arrival times of these packets. While later studies call it "packet stream" or "packet chirp" [140], we believe they are still in the same category.

An earlier work using packet trains is BPROBE and CPROBE [23], which measure the bottleneck bandwidth and the available bandwidth respectively. The idea to find the bottleneck bandwidth in BPROBE is to send a series of ICMP echo packets to the server and make interpretations based on the inter-arrival times of the incoming responses. The authors advocate that filtering and clustering methods can help to remove queueing or cross-traffic related noise. Similarly CPROBE is based on the inter-arrival of ICMP responses, however, it only filters the unusally long and short inter-arrival values to remove extreme results, due to context switchs at the sender, and estimates the available bandwidth by the arrival time difference between the first and last ICMP responses, which includes the queueing and cross-traffic. Also, these tools typically use 400-1200 probes to have measurement results for the bottleneck links ranging from 56Kbps to 10Mbps. Again, high speed links require better accuracy on sending times.

A widely-recognized tool in estimating the available bandwidth is "Pathload" [81] with the Self-Loading Periodic Streams (SLoPS) concept, which creates an instantenous congestion and thus queueing along the path, to see if the sending rate exceeds the available bandwidth by the inter-arrival times of the packets in stream. Pathload periodically sends such streams to adaptively converge to a minimum and maximum bound for the available bandwidth. The authors refer to this set of streams as "fleet" and typically use 12 fleets each having 10 streams, where a stream itself has 100 packets, making up 12000 packets for a single measurement. Thus, Pathload consumes considerable bandwidth during measurement tests and having many users making such tests may create problems within the network. For example, in Spruce [168] tests, Pathload produces 2.5-10MB probe traffic per measurement whereas Spruce produced only 300KB. Also, Pathload usually converges in 20 seconds.

Another popular tool is “PathChirp” [148] using essentially the same technique as Pathload but with an exponential space within packets of a “chirp” to find out more information about the available bandwidth in a single iteration. Pathchirp and Pathload have similar accuracy in tests [38, 163]; typically an accuracy within 10-20% range. However, Pathchirp consumes significantly less bandwidth and converges quickly. For example, a study [163] shows PathChirp consumes 0.2% of the available bandwidth on the GigE link while Pathload uses 3-30% of the available bandwidth.

Although some authors [87] claim that packet-pair approach is more resistant to noise than packet-train approach, results [163, 38] show that packet train/stream/chirp based approaches are more robust than packet-pair approaches.

Other techniques

A popular, widely-recognized tool is Iperf [58] which finds the available bandwidth by literally filling the capacity with its TCP or UDP traffic. Iperf provides stable results though consuming most of the available bandwidth (a study shows 75% in [163]) for each measurement is not desirable. Also, when used with TCP, more than 1% packet loss causes Iperf to underestimate the available bandwidth due to the nature of TCP.

In order to detect end-to-end packet loss-rate in both forward and reverse paths, TCP-STING [158] sends a series of TCP packets to the destination and relies on the prompt TCP acknowledgements for each packet in order to interpret the packet loss in both ways. Because TCP normally delays acknowledgements and other similar issues, the paper’s implementation has several workarounds to have this method working, such as sending always out-of-order packets so that the destination responds with a timely acknowledgement for each packet. The measurement results conducted with webservers also revealed that on average, the reverse path loss rate is more than 10 times greater than the forward path loss rate. The method also has the advantage that it does not require any modification at the destination.

While all of the techniques discussed so far require no change in the intermediate nodes of the network, some additional work [102, 105] has been conducted to improve the measurement accuracy and capabilities by adding support to routers in the network. For example IPMP (IP Measurement Protocol) [102] proposes a new measurement protocol on top of IP, as an alternative to ICMP, to have routers add timestamps and path information to incoming packets, advocating that this can be efficiently implemented. The protocol also supports “information response” messages to inform an end node about the accuracy of the router’s time so that the endpoints can determine whether the timestamp value provided is worth considering for measurement or not. The authors claim that such a capability on even some of the routers may make a big difference to better understand the network experience of the packets. While this capability is not available in today’s routers, having such a protocol at borders of network service providers can help understanding the delays packets encounter along the path. One must also take into account the time synchronization issues as well as different treatment of IPMP packets.

Having finished our discussion on the available technologies, in the next chapter, we describe the ChoiceNet architecture and then propose our verification service architecture.

Chapter 3

ChoiceNet and Verification

3.1 ChoiceNet Overview

3.1.1 Architectural Design Context

The Internet is a large system, with multiple providers of equipment and services, with many entities playing many roles, some of which mirror others partially or wholly. It has become clear in recent decades that the tremendous success of the Internet is due to the possibility of many viable value propositions to exist in this space, such that many different businesses can profitably provide parts of the overall functionality. In doing so, often the interests of the various players come in conflict with each other, though at the highest level they may be in a symbiotic relationship; in [36], the authors call these "tussles in cyberspace", and consider the function of architecture to be to provide the space in which such necessary tussles may proceed, without restricting or biasing the outcome.

In functional terms, the purpose of architectural design is to articulate the necessary entity interactions required for the desired functioning of the whole system. The entity roles may be thought of *containers* of functionality, which are specified by the architecture. Various roles may be fulfilled by different business entities. The *mechanism* that each participant decides to realize within their box may be different, but the *interactions* between the various boxes must transcend the choice of any particular participant, so must be specified by the architecture, and the *interfaces* between the boxes to enable such interactions must be standardized, as shown in Figure 3.1. Thus the identification of the roles and the interfaces constitute what Clark et al. have referred to as the "cut-points of the architecture".

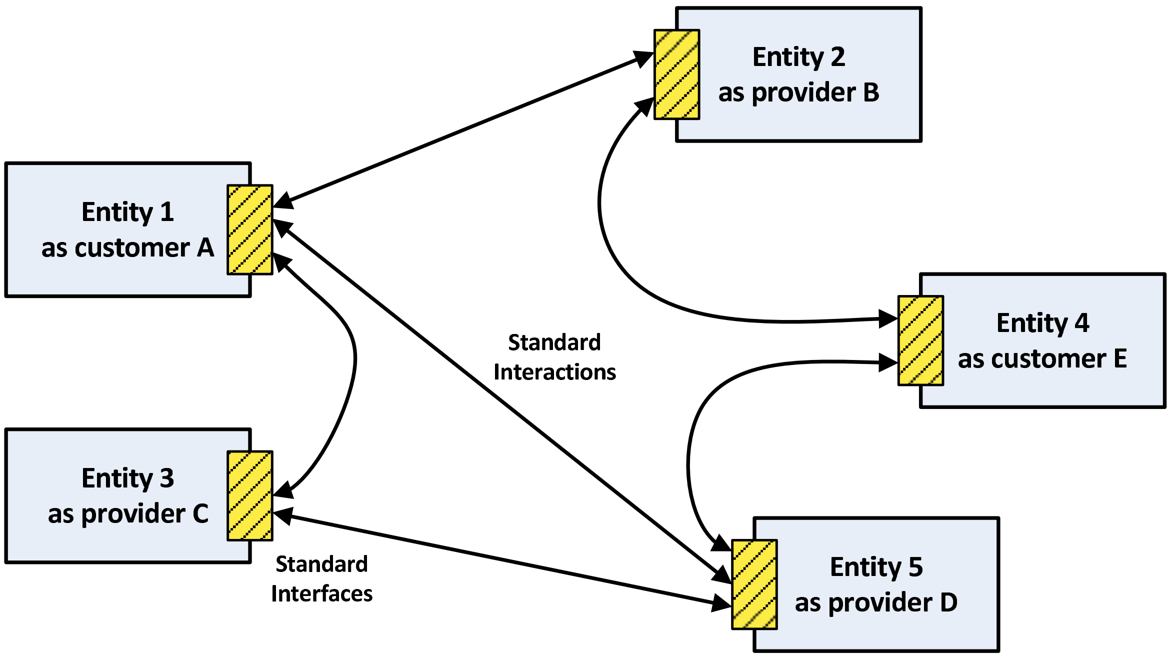


Figure 3.1: General Architecture Design. Each blue box represents an entity and the yellow hatched rectangles represent the standard interfaces to enable standard interactions.

For a diverse, collaborative and dynamic environment such as the Internet, it is necessary for the architecture to be minimal; each role must be postulated only when the absence of any entity in that role would leave some essential function unfulfilled in the ecosystem. The roles must also be specified only in terms of their interactions, not the mechanism that any participant may use to realize the role. On the other hand, the interactions must be well-articulated and well-defined to easily facilitate each role to meet the standards, and avoid compatibility issues. The goal of the architecture, then, is *to identify a minimal set of roles and interfaces amongst them, such that roles that are likely to be filled by different business entities in the real world are separated, there is a value proposition for each role, and their interactions are clearly defined and specified.*

In the next subsection, we describe the broader architectural vision of ChoiceNet. We then describe the verification service architecture in Section 3.2, keeping in mind that the architecture must not mandate heavyweight interactions that themselves impose a large overhead of communication on the network (though specific realizations may do so). It is important to note that a particular business may fulfill more than one of the roles below. Similarly, particular business entities may carry out the mandated interactions using the interfaces we specify below, or other mechanisms that they agree on, even paper contracts or phone calls (as is the case for similar interactions today). However, standardizing such interactions in the architecture (by attempting an adequate specification of the interfaces and interactions) would allow businesses that do not already have existing business relationships with other parties to start interacting with others. Hopefully this facilitates the entry of innovators into the marketplace and allows them to quickly build up their ecosystem, by providing a ready framework that lowers the barrier.

3.1.2 ChoiceNet Architecture

The success of the current Internet is driven very largely by the economic interaction of many entities, acting as customers and providers of various hardware and software services for network functions and computing and storage functions, all mediated by economic interactions. However, this economic reality is not represented in any of the network software or protocol interactions, thus the economic incentive for providing innovative services is detached from the network interaction.

No valuable network function can be assumed to be free, or made available as a public service. In monetary transactions for services, competition improves the collective benefit. Competition is more effective when the marketplace is larger. A small marketplace can become a buyer's market or seller's market more easily. A larger variety of buyers means a larger variety of producers and sellers can be sustained. In particular, small producers, who in a marketplace of only a few large buyers will either become captive to one of them or be driven out, may flourish in a large variegated marketplace containing small buyers who can be directly sold to. These reflections prompt the goal of ChoiceNet to introduce architectural entities into the Internet to enable fine-grain economic interactions.

In order to combine the incentives for new alternatives with the competition in a market driven environment, ChoiceNet is based on three principles [151]; (i) "Enable Alternatives": provide the mechanisms and building blocks for users to be presented

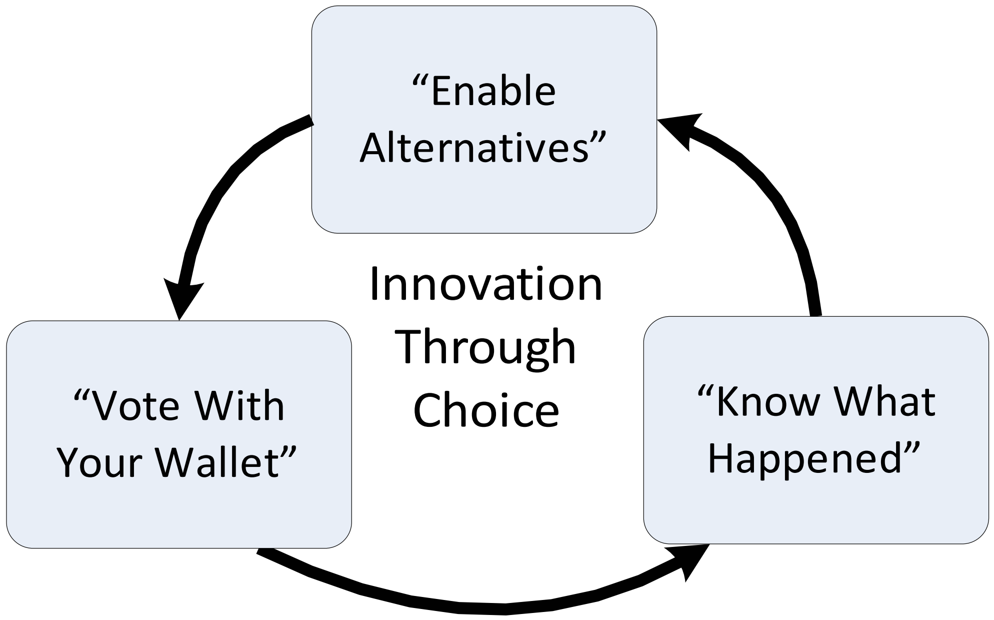


Figure 3.2: The ChoiceNet Cycle

with, and to choose different services easily, (ii) "Vote with your wallet" to provide the means for secure, scalable and fine-granularity payment protocols to allow the user to promote better performing providers, (iii) "Know what happened": allow the user to find out which provider to blame if service expectations are not met. These principles constitute a cycle as shown in Figure 3.2 and are essential for each other, While this cycle indeed exists in real-life, the goal of ChoiceNet is to provide the mechanisms to perform this cycle in much shorter time scales, at finer granularity, and largely automatically to boost the competition necessary for innovation. We should also emphasize that payment being an important part of the architecture does not imply additional costs for users, instead inexpensive payment models [65] can be utilized.

The ChoiceNet architecture provides a common platform for service providers to advertise their services and for customers (also called "choosers") to easily discover, negotiate and pay for these services. The component empowering the advertisements for services is called the "marketplace" where service providers register their services and customers discover them via querying the marketplace. Once a customer decides the service to purchase, further ChoiceNet interaction between customer and provider creates a contract, with the customer receiving a *token* of some form. These interactions constitute what we term the “Economy Plane” of ChoiceNet, and are all paralleled by real-world interactions that take place, but outside the network architecture, today. Subsequent to this, the customer may use the token to obtain the desired service in the same manner as would take place in the current Internet - we refer to this as the “Use Plane”. Such services could be for endpoint services, path or pathlet services, or even in-network processing services. However, in the ChoiceNet view, the customer would undertake individual contracts with each of the providers of such services along the complete path to compose the entire service, and would be financially rewarding each provider, thus creating incentive for innovation at all such service points, endpoints, paths, and intermediate processing/storage services. In reality, automated software acting on behalf of the user would undertake these decisions, translating high-level goals of user experience into low-level ones.

ChoiceNet interactions are enabled through two types of interfaces [151]; (i) customer/provider interface in the economy plane and (ii) client/service interface in the use plane. The business relationships such as service advertisement, contract setup, payment or identity use the customer/provider interfaces, whereas technical service interactions to setup and provide the service use the client/service interface. Because a service is defined as “any functionality performed in the network” including combination of small services, ChoiceNet’s interfaces enable entities to realize complex service models where an entity can act both as a provider to some customers and also at the same time act as a customer to some providers. Figure 3.3 illustrates such a case where a “bundle provider”, or simply a bundler, creates a path service by stitching path services from different network service providers (NSP). The bundler first makes contracts with each NSP and then resells the combined path service to customers in the economy-plane. Making the contract, the bundled path service is provided to the customer in the use-plane. Realistically, composing a set of services advertised in the marketplace to realize a complete service for the customer will be a complex task for all but the simplest cases, thus the task must be automated and performed by software agents. Providing such software may itself be a service in the ChoiceNet framework. Furthermore, the advertisements in the marketplace must be semantically enriched to allow automated composition, thus marketplaces are more like ontologies than directories. We do not discuss these ChoiceNet issues in further detail in this thesis.

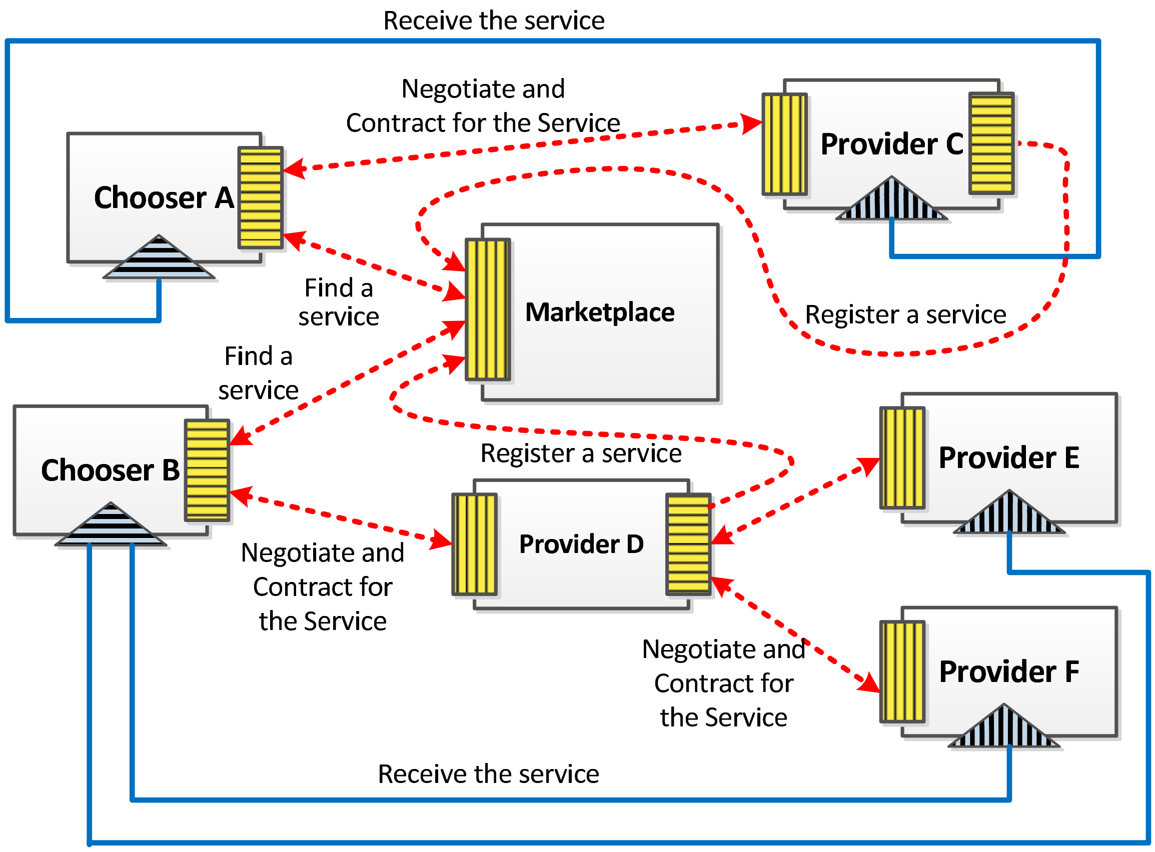


Figure 3.3: Entities and interactions of the ChoiceNet Architecture

Figure 3.3 shows the ChoiceNet Architecture, where the red dashed arrows represent the economy-plane interactions using the ChoiceNet interfaces as yellow hatched rectangles, and the blue solid lines represent the use-plane interactions using the interfaces shown as triangles. Providers “register their services” to the marketplace so that choosers can first “query these services” and then negotiate with the providers in order to “receive the service”. Also, horizontally hatched interfaces denote the chooser interfaces, whereas vertically hatched interfaces denote the provider interface. Additionally, the architecture allows a provider D to make contracts with providers E and F, in order to bundle their services and resell them to the customer. Note that the blue solid lines do not have a specific direction because the received service can be in either direction such as video streaming (to the chooser) or storage (from the chooser).

While ChoiceNet empowers the user to make fine-grain choices and reward providers who perform well with continued contracts, the user must also be able to discern which service provider is to blame if the user's overall satisfaction metric is not met. If a user is not satisfied with the quality of experience in viewing streaming video, they need to know if the fault lies with the player or other local software, their home network, their home Internet service, any of the several provider hops their video traffic passes through in sequence, or the video streaming service at the far end. This will generally require measurements to be made inside the network cloud. Since the individual consumer does not typically have presence except at the endpoints, this points to the potential of a specific class of service provider roles, independent third-party verifiers, and the need of architectural mechanisms specific to the collection and providing of such verification information. In the next section, we discuss the specific goals and design decisions along with the role, interaction and interface descriptions.

3.2 Verification Service Architecture

In this section, we first describe the roles of the Verification Service Architecture [8], followed by the interfaces and interactions between these roles. We acknowledge that there are many abbreviations used in our work to refer to existing technologies as well as to describe the components of our architecture, which may reduce readability. Therefore, Table 1 provides the meaning of abbreviations.

3.2.1 Architectural Roles

Figure 3.4 shows the roles of our verification architecture, where the dashed red arrows represent the service setup interactions (economy plane interface as yellow rectangles) and the green dotted arrows represent the measurement data transfer (use-plane). In addition, the thick blue solid line represent the service received by the chooser from SP (use-plane interface as triangles). Moreover, horizontally and vertically dashed interfaces symbolizes the ChoiceNet customer and provider interfaces respectively. Also, rectangles with “S” and “R” denote the MSP sender and receiver interfaces respectively. Because MSP1 and MSP2 have very similar structures, MSP2 is represented as a box.

Measurement Service Provider (MSP): As the core role of the verification service architecture, an MSP is an independent provider responsible for producing and delivering

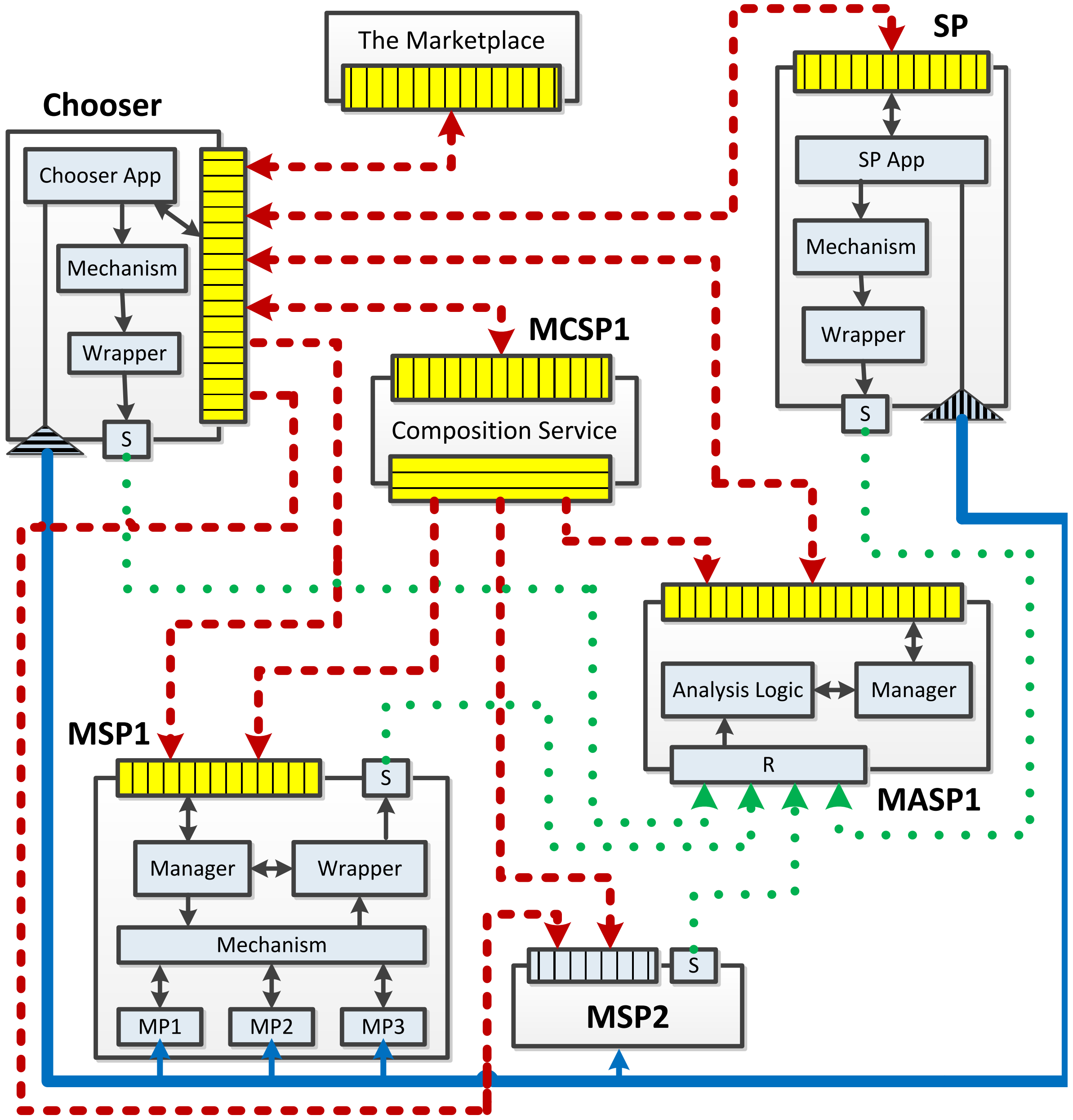


Figure 3.4: Verification Architecture Roles

the measurement data (MD) relevant to the chooser. The role itself consists of several sub-entities as shown in Figure 3.4. An MSP utilizes one or more "Measurement Points (MP)" to produce the MD. Following the PerfSONAR (discussed in Section 2.2.2) terminology, an MP is the actual measurement equipment located inside the network, preferably at a point of presence or Internet Exchange Point (IXP), typically with passive measurement capability to capture chooser packets. Deploying MPs at IXPs is attractive because this requires no extra support or trust from any network service provider and yet the system can have access to the ingress and egress traffic of hundreds of providers [2] at a single location.

Once a packet is captured, the production of MD depends on the selected "MSP Mechanism (MM)". An MSP can offer several such MMs as part of the service advertisement, and the same MM may need to be used at multiple MSPs to compose a verification service. Choosers might prefer different MMs for different applications and several MMs may run concurrently. Although MP and MM are logically separate entities, due to performance reasons, it is suitable to have them as different processes in the same device, or even have MMs executed as simple functions in MP. A simple MM example can be a "throughput calculation mechanism", which calculates the number of bytes received within a second for the desired flow, but more complex mechanisms can be envisaged as shown in Section 4.3.

As part of low-level detail abstraction, An MP never directly interacts with the customer. Instead, an "MSP Wrapper" entity provides the MD in a "service-specific" representation. Because minimizing the network load caused by MSP is essential and because delivering large amounts of MD over ChoiceNet's extensible (for example XML-based) messaging language can waste bandwidth significantly, we introduce an optional verification service specific MD transmission interface, called "MSP Interface", at sender as "S" interface and at the receiver as "R" interface respectively. Although usage of these "MSP Interfaces" is not required, it provides the efficiency we need in practice. The type and format of the messages exchanged by this MSP Interface are negotiated by the ChoiceNet Interface, including a possible temporary ID and encryption key to maintain secure communication. In addition, thousands of customers may demand this service from the same MSP and in order to achieve scalability, MSP Wrapper and MSP Interface functionality may be realized by several machines, if necessary.

In Figure 3.4, each role also has a "manager" sub-entity, responsible for general administrative tasks of the provider such as configuration, logging, payment, identity, registry or advertisements. These tasks are vital for each provider and that is why we keep this sub-entity in each provider. However, it is a per provider sub-entity so a provider which has several roles can use only a single manager to do all such tasks. It can also act as a central entity forwarding user requests to appropriate entities. Also in Figure 3.4, each role has at least one ChoiceNet interface, because any provider advertising itself as an independent provider must have a ChoiceNet interface. However, a large provider who has multi-roles can prefer offering only bundled services and in this case, entities of different roles can communicate by ad-hoc means since they are indeed part of the same provider and thus they do not need separate discovery, negotiation or payment. It is up to the chooser to choose between the flexibility in mixing and matching these capabilities, and the convenience (but also opacity) of a bundled service, possibly with different price tags.

Measurement Composition Service Provider (MCSP): In the current Internet, a user flow typically goes through more than one network service provider (NSP) before reaching its destination, and therefore there may be several MSP choices to measure the flow. Having several such MSP alternatives with various capabilities to choose from, makes verification service composition a challenge by itself. Therefore an MCSP role makes such service compositions convenient for customers. An MCSP first negotiates with various MSPs to setup the desired measurement service with appropriate metrics, mechanisms, MD representation. MCSP then finds an appropriate analyzer (describe below) for verification. Having found the entities, MCSP returns a "recipe" to the customer. Having got the recipe, the customer can now communicate with the providers to make a service contract. These negotiations may also include the specifications of MSP interfaces such as message format, endpoints, temporary encryption keys etc.

It is important to note that MCSP is a special case of service composition, a general feature of ChoiceNet. That is, it can be realized as part of a general ChoiceNet service composition role, instead of a separate entity in Figure 3.4. Also, it may very well be implemented as a software in customer's computer, or alternatively, as part of a large provider (say provider X) which also has MSP functionality. The last scenario allows "MSP peering", where such a provider X can contract with another MSP, in a ChoiceNet customer/provider fashion, to broaden its measurement capability geographically and to offer advertisements especially desirable for customers communicating over long distances. In that case, a customer may only be negotiating with provider X which may indeed have multiple providers performing as subcontractors. Furthermore, a hierarchy of MSPs is also possible, where a provider X may have contracts with many geographically distant small MSPs to increase its competitiveness. We anticipate our work is to design the architecture enabling such possible scenarios and let the market decide for feasible practices.

Measurement Analysis Service Provider (MASP): The raw MD itself is not generally useful to the customer until it is analyzed, which makes MASP a crucial role for verification. Having MD as input, MASP typically applies statistical methods (similar to the measurement tools in Section 2.2.4) to provide useful information to the customers. However, we anticipate a variety of "detail levels" can be desirable for distinct customers and their applications. While some applications or users may only ask for a "yes/no" reply for each service provider, others might want more in-depth analysis. Therefore MASP should support various levels of detail. Additionaly, as part of service negotiation, a MASP can optionally indicate a receiver, "R", MSP Interface for each sender "S" MSP Interface that it expects receiving MD from. Similar to MCSP above, a MASP can also be implemented as a software in customer's computer or as a part of a general analysis provider, or alternatively as part of a large measurement provider that has MSP or MCSP.

Chooser and SP: In some cases (for example, dynamically changing throughput measurement, described in Section 4.3), a MASP may require the MD input from the chooser and SP (a service provider that users get end-point service such as video streaming) in order to compare them with MD collected from MSPs. In that case, chooser and SP should also run a measurement mechanism (MM), possibly as a plug-in.

Although we usually expect applications to "know" what measurement metrics they are looking for, some applications may optionally delegate this task entirely to a "Measurement Metric Converter Service (MMCS)" (not shown in Figure 3.4) which converts a high-level statement into low-level measurement metrics. For example, a VoIP application, may utilize ChoiceNet Interface with a "VoIP" as metric, which MMCS converts to, say <80ms OWD (one-way delay), <5ms jitter and >200Kbps throughput for 100ms intervals, i.e. a mapping from high-level to low-level goals. Furthermore, the architecture does not prohibit any 3rd party agents that can act on behalf of users (not shown in Figure 3.4), especially for computers with low processing power or for corporations with several computers, as long as agents speak the ChoiceNet language.

3.2.2 Architectural Interfaces

Having described the roles, we now turn our attention to the interfaces shown in Figure 3.4, which play a critical role in ensuring that customers and providers understand each other. Because setting up a verification service involves several stages, we first explain such stages, followed by the general information exchange structures and the set of interactions to perform the service. It is important to note that the architectural roles we discussed earlier, also use the native ChoiceNet services and thus their interactions, such as marketplace queries, payment transactions, authentication messages. For example, a MCSP may require marketplace queries and a MASP or MSP may require advertising their capabilities. Furthermore, all such roles may require payment and authentication for their services. However, these interactions are not the focus of our work so we prefer explaining the information exchanges only related to our work.

Figure 3.5 represents a sequence of events, in a distributed system in a flow chart fashion. We emphasize that it is not an algorithm performed by a single entity, instead the state diagrams for each entity is shown in Figure 3.7, 3.8, 3.9 for MASP, MSP and the chooser respectively. The chooser (or an application on behalf) must first determine the desired verification service followed by its endpoint requirements. In order to achieve this, the customer (or application on behalf) may have the built-in logic to select such a service, or may utilize a MMCS service, described earlier. If the mechanism (MM) requires support from the destination (SP) endpoint, the customer then negotiates with the SP about the availability of such a capability. If it does not work, then an alternative MM can be tried, perhaps with a higher cost or less accuracy. Assuming a common MM between endpoints is found (or not needed at all), the customer then searches the marketplace a MCSP, and upon finding one, the customer now requests a “recipe” from this MCSP, where the “recipe” is a list of the providers to realize the verification service. MCSP searches either a marketplace or its well-known provider list to come up with a “good match” based on the customer’s needs, and returns a recipe, if possible. If MCSP fails to return a recipe, then the customer can simply request the composition service from an alternative MCSP. Assuming a recipe is returned, the customer now makes individial contracts with the providers in the recipe. The first contract should be with MASP because it in turn provides the measurement interface that other nodes connect to. Having contract with MSPs and SP, each provider now checks with MASP to ensure that the connection and the credentials are working. Upon setting up the resources, each provider sends a "ready" message to the chooser, and once all expected ready messages are received, the chooser starts using the use-plane. This, in turn, causes MD to be produced by MSPs and be collected by MASP. MASP analyzes the MD and returns the verification results to the customer. Based on these results, customer now can make choices accordingly. We also emphasize that failures during these stages typically cause customers to switch to a different mechanism or provider and "too many" such failures can cause the customer to quit using a service or a provider.

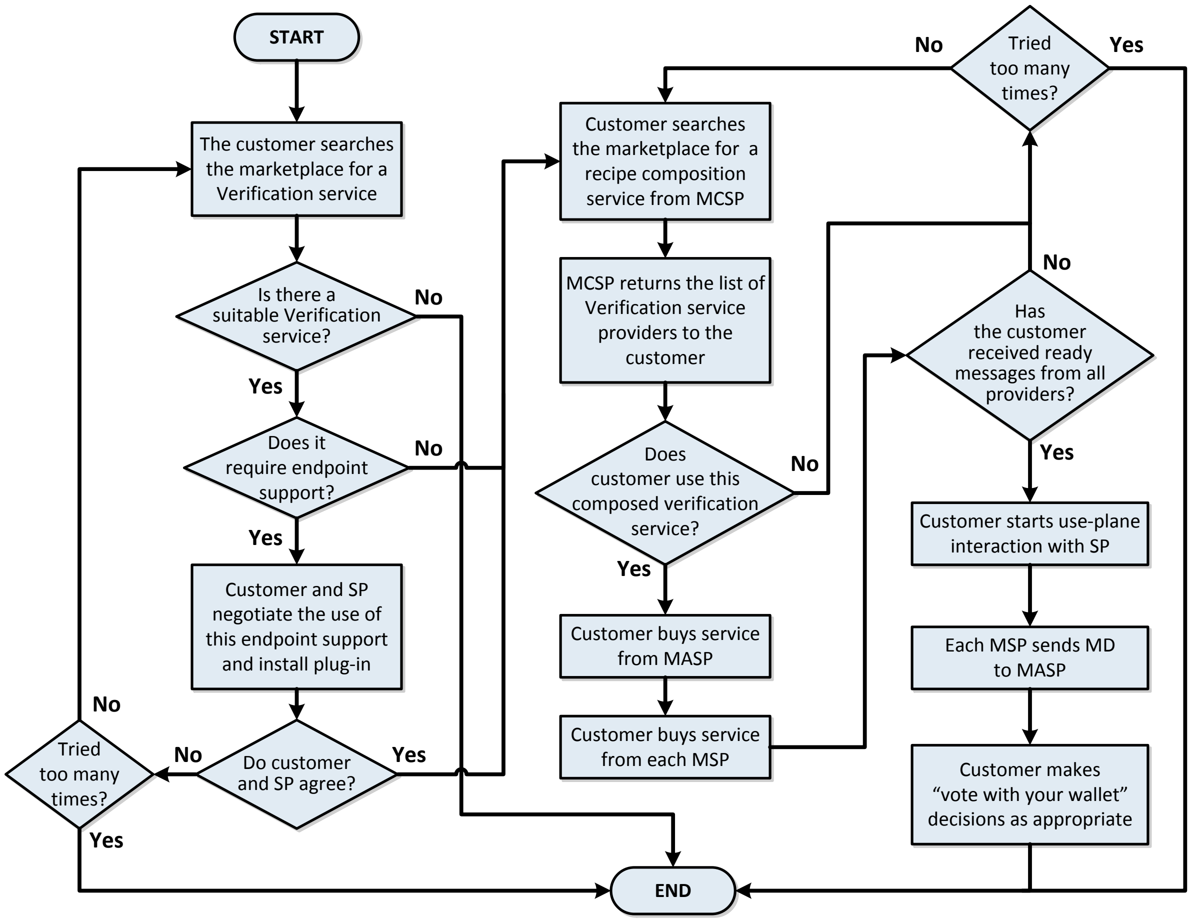

START
The customer searches the marketplace for a Verification service
Is there a suitable Verification service?
No
Yes
Does it require endpoint support?
No
Yes
Customer and SP negotiate the use of this endpoint support and install plug-in
Do customer and SP agree?
No
Yes
Tried too many times?
No
Yes
Customer searches the marketplace for a recipe composition service from MCSP
MCSP returns the list of Verification service providers to the customer
Does customer use this composed verification service?
No
Yes
Customer buys service from MASP
Customer buys service from each MSP
Tried too many times?
No
Yes
Has the customer received ready messages from all providers?
No
Yes
Customer starts use-plane interaction with SP
Each MSP sends MD to MASP
Customer makes "vote with your wallet" decisions as appropriate
END

Figure 3.5: Verification service as a sequence of events in a distributed system

Before going into the interaction details, we first give a high-level view of the information exchanged between entity roles. We emphasize that the following simplified

XML-based structures below are presented only to give readers a better understanding of the verification service specific information exchanged by different parties, rather than exhaustively showing all possible messages.

Flow: Listing 3.1 defines a “flow” by the endpoints, where endpoints are represented by the five tuple (source IP address, source transport port, destination IP address, destination transport port and the transport protocol). While this is the most common way to define a flow, the extensible nature of the architecture allows different flow definitions, without changing the architecture itself. For example, IP version 6 can easily be defined by only changing the version attribute value and a new transport protocol can similarly be defined only by modifying the port attribute. In fact, the architecture even allows aggregated flows, such as flows based on MPLS or VLAN tags or IP address prefixes. Furthermore, we envision that this extensibility facilitates compatibility with Future Network Architectures (discussed in Appendix 2.1).

Listing 3.1: Flow definition

```xml
<flow>
 <endpoint type="source">
   <ip version="4">chooser IP</ip>
   <port type="tcp">chooser port</port>
 </endpoint>
 <endpoint type="destination">
   <ip version="4">service provider IP</ip>
   <port type="tcp">service provider port</port>
 </endpoint>
</flow>
```

Path: Listing 3.2 defines a “path” by the endpoints and a number of service providers in between. As well as the endpoint definition discussed as part of the flow definition, ingress and engress can be defined by protocols other than IP, if negotiating parties agree.

Listing 3.2: Path definition

```xml
<path>
 <endpoint type="source">
  <ip version="4">chooser IP</ip>
  <port type="tcp">chooser port</port>
 </endpoint>
 <nsp position="1" id="nsp1">
```

```xml
     <ingress><ip version="4">NSP1 IP1</ip></ingress>
     <egress><ip version="4">NSP1 IP2</ip></egress>
    </nsp>
    <nsp position="2" id="nsp2">
     <ingress><ip version="4">NSP2 IP3</ip></ingress>
     <egress><ip version="4">NSP2 IP4</ip></egress>
    </nsp>
    <nsp position="3" id="nsp3">
     <ingress><ip version="4">NSP3 IP5</ip></ingress>
     <egress><ip version="4">NSP3 IP6</ip></egress>
    </nsp>
    <endpoint type="destination">
      <ip version="4">service provider IP</ip>
      <port type="tcp">service provider port</port>
    </endpoint>
   </path>
```

Metrics: As shown in Listing 3.3, metrics can be used to define a constraint or as part of a measurement request.

Listing 3.3: Metric definition for analysis and measurement

```xml
<!-- metric definition for analysis -->
<metrics>
 <metric type="constraint" name="one-way-delay">
  <value type="time" condition="less-than" unit="ms">80</value>
 </metric>
 <metric type="constraint" name="throughput">
  <value type="bits" condition="greater-than" unit="mega">2</value>
 </metric>
</metrics>
<!-- metric definition for measurement -->
<metrics>
 <metric type="measurement" name="one-way-delay"/>
 <metric type="measurement" name="throughput"/>
</metrics>
```

Mechanism: Listing 3.4 defines a mechanism by its name and input parameters to be used. In this XML snippet, the name attribute is a highly simplified, whereas in reality, a mechanism can be defined by its URL and XML Namespace for clarification. The inner XML tags represent the parameters of this mechanism and it simply means a "marker" packet is sent inline (not as individial packets) for every 100 packets sent. The details of "marker mechanism" is given in Section 4.3.

Listing 3.4: Mechanism definition

```xml
<mechanism name="marker" version="1.0">
```

```
    <parameters>
     <parameter name="rate" type="count">100</rate>
     <parameter name="packet" type="inline" content="application"/>
    <parameters>
   </mechanism>
```

Wrapper: A standard MD representation is essential for 3rd parties to interpret the meaning of MD and this is defined as "wrapper" in Listing 3.5, where "measurementFormat" represents a well-known standard output format, meaning the wrapped messages will have 300 bytes and will be sent for each MD message upon receipt.

Listing 3.5: Wrapper definition

```
   <wrapper name="wrapperDefinition" version="1.0">
    <format name="measurementFormat" version="1.0" messageSize="300" unit="
        bytes"/>
    <frequency type="uponReceipt"/>
   </wrapper>
```

Analysis: The results of the analysis must be easily consumed by the customers regardless of the MM, MD format. The "analysisFormat" in Listing 3.6 refers to a minimum report format and specifying that it will be sent to the customer every second.

Listing 3.6: Analysis definition

```
   <analysis name="analysisDefinition" version="1.0">
    <format name="analysisFormat" version="1.0" detail="minimum" granularity="
        provider"/>
    <frequency type="time" unit="second" value="1"/>
   </analysis>
```

Preference: The preference format in Listing 3.7 defines the criterias of the customer in a request to a MCSP. In this case, the customer's preference is a single result with the lowest price.

Listing 3.7: Preference definition

```
   <preference type="price" value="lowest" numberOfResults="1">
```

In addition to the basic structures we presented above, other definitions will be provided when needed, but now we describe the interactions that utilize these snippets.

The general case where a chooser utilizes ChoiceNet Interface to compose, make contracts and then receive the verification service is shown in Figure 3.6, and below we describe each interaction in detail by referring the messages in this figure.

Chooser to Marketplace, SP and MCSP: As shown in Figure 3.5, assuming chooser application knows what metrics to measure; it first searches for a verification service from the marketplace as shown in Message 1 in Figure 3.6. If this service requires a mechanism that needs SP and chooser support, then the chooser negotiates with SP to request this support as shown in Message 2 in in Figure 3.6. Note that the chooser may also install a plug-in mechanism to realize this service. Having a successful negotiation with a “decision accept” message as shown in Listing 3.8, the chooser again searches the marketplace for recipe composition service and marketplace replies with a MCSP, as shown in Message 3 in Figure 3.6. Note that this step can be combined with Message 1 as well. Having agreement with SP and learning about MCSP, the chooser sends “flow”, “path”, “metric” with constrainsts, “mechanism”, “wrapper”, “analysis” and “preference” information with values in Listings 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7 respectively, represented by Message 4 in Figure 3.6, and the next set of interactions are described in the following paragraph.

MCSP to MSPs, MASP and Chooser: Assuming MSP1 is in between NSP1 and NSP2, and MSP2 is in between NSP2 and NSP3, MCSP negotiates with MSP1 and MSP2 by sending the “flow”, “path”, “metric” for measurement, “mechanism”, “wrapper” information with values in Listings 3.1, 3.2, 3.3, 3.4, 3.5 respectively, represented by Message 5 and 6 in Figure 3.6. If MSP1 can provide this service, it sends back an “decision accept” message as in Listing 3.8 to MCSP and the same goes for MSP2. MCSP then sends the “flow”, “path”, “metric” for analysis , “mechanism”, “wrapper”, analysis” information to MASP1, as shown in Message 7 in Figure 3.6. Assuming all entity roles accept the request, as shown in Message 8 in Figure 3.6, MCSP now returns a “decision recipe” to the chooser as in Listing 3.8.

Listing 3.8: Decision accept and recipe definitions

```
<!-- decision accept definition -->
<decision type="response" accept="yes">

<!-- decision recipe definition -->
<decision type="response" accept="yes" recipe="yes">
```

```
    <provider position="1" type="msp" id="msp1">
     <interface type="choiceNet" version="1.0">
      <ip version="4">MSP1 IP</ip>
      <port type="tcp">MSP1 TCP Port</port>
     </interface>
    </provider>
    <provider position="2" type="msp" id="msp2">
     <interface type="choiceNet" version="1.0">
      <ip version="4">MSP2 IP</ip>
      <port type="tcp">MSP2 TCP Port</port>
     </interface>
    </provider>
    <provider type="masp" id="masp1">
     <interface type="choiceNet" version="1.0">
      <ip version="4">MASP1 IP</ip>
      <port type="tcp">MASP1 TCP Port</port>
     </interface>
    </provider>
   </decision>
```

Chooser to MASP and MSPs: Having a recipe, the chooser now makes individual contracts with MASP, SP, MSPs. While these interactions are similar to the ones MCSP has done previously, these are separately necessary to make contracts with each provider. The first provider to make contract is the MASP because it will provide the interface that other providers need to send MD to. The chooser message includes the “flow”, “path”, “metric” for analysis, “mechanism”, “wrapper”, “analysis” as shown in Message 9 in Figure 3.6. Assuming MASP1 replies with an “decision accept” and after chooser makes a payment and MASP1 replies with an “MSP Interface” in Listing 3.9, accompanied by credentials for each entity. That is precisely the reason why Message 9 includes 4 Messages. In this scenario, there are four such ID-KEY pairs needed; chooser, MSP1, MSP2, SP respectively.

Listing 3.9: Interface definition with temporary credentials

```
   <interface type="measurement" version="1.0">
      <ip version="4">MASP1 IP</ip>
    <port type="tcp">MASP1 TCP Port</port>
    <credential type="temporary">
     <user>abcd</user>
     <password>1234</password>
     <ip required="yes" version="4">MSP1 IP</ip>
    </credential>
   </interface>
```

The chooser now sends these ID-KEY pairs to appropriate providers as shown in Messages 10, 13 and 16 in Figure 3.6. Having their internal setup done, all roles try to connect to MASP1 with these credentials as represented in Message 11, 14, 17 in Figure 3.6. Having succeeded, each will send a "ready" message in Listing 3.10 to the chooser as shown in Message 12, 15, 18 in Figure 3.6. Note that the exact order of these messages does not matter, but once all "ready" messages are received, it signals to the chooser that the system is now ready for verification service.

Chooser, MSP and SP to MASP: The chooser starts use-plane interaction, and in this scenario, we are sending packets from the chooser to SP. Chooser, MSP1, MSP2 and SP now sends MDs to MASP1 as shown in Messages 19, 20, 21, 22 in Figure 3.6 respectively. Listing 3.10 shows a sample MD based on the marker mechanism with incomplete information such as hardware or software tags. However, as discussed in Section 3.2.3, creating a complete measurement specification is beyond the scope of this thesis.

Listing 3.10: A sample measurement data definition

```
<md name="markerPacketMeasurementData" version="1.0">
 <experiment id="45632fd">
  <starttime>546343432743</starttime>
  <endtime>546343492674</endtime>
 </experiment>
 <node id="5472473453">
  <domain>...</domain>
  <hardware>...</hardware>
  <software>...</software>
 </node>
 <mechanism name="marker" version="1.0">
  <parameters>
   <parameter name="rate" type="count">100</rate>
   <parameter name="packet" type="inline" content="application"/>
  <parameters>
 </mechanism>

 <element type="sequence">
  <value>57</value>
 </element>
 <element type="receivedBytes">
  <value type="bytes" >58460</value>
 </element>
 <element type="captureTime">
  <value type="time" relative="yes">565</value>
 </element>
 <element type="previousTime">
```

```
      <value type="time" relative="yes">535</value>
     </element>
   </md>
```

MASP to Chooser: As an example, assuming that all MD collected are similar to one in Listing 3.10 means that NSP2 is creating a significant latency and therefore it is the one to blame for the one way delay performance problems. MASP1 now provides a sample output to the chooser as shown in Message 23 in Figure 3.6 and it is defined by the "result for verification" message in Listing 3.11. Note that even though chooser application has asked for 2Mbps throughput, it actually did not send that much traffic, therefore we can not blame any provider that 2Mbps throughput performance is not met.

Listing 3.11: Result definition for service setup and verification

```
<!-- result definition for service setup -->
<result name="serviceResult" version="1.0">ready</result>

<!-- result definition for verification -->
<result name="measurementResult" version="1.0">
 <provider type="nsp" id="nsp1" owd="yes" thr="yes"/>
 <provider type="nsp" id="nsp2" owd="no" thr="yes"/>
 <provider type="nsp" id="nsp3" owd="yes" thr="yes"/>
</result>
```

3.2.3 Measurement Specifications

The measurement data is the building block to analyze and verify the service agreements. Because different services may have different promises, various metrics may need to be represented by the measurement data such as delay, throughput, packet loss. A common agreement on the exact definition of measurement metrics can be achieved for human readers by utilizing the existing studies such as the IP Performance Metrics (IPPM) [134, 40] for delay and jitter definitions, and the study in [140] for bandwidth concepts (link capacity definition as well as tight and narrow link definitions). In the measurement literature discussed in Chapter 2, PerfSONAR measurement framework provides a structural and extensible XML based format [137] to represent various measurements.

Another important ongoing work as part of GENI testbed project (described in Appendix B.1) is the GEMINI [62] and GIMI projects [64]. The primary goal of these projects is to provide a system that can easily collect and visualize measurement data for the experimenters using GENI. In order to realize such a system, they aim to represent and archive the measurement data produced by the experiments on the GENI testbed, by working on MDOD (Measurement Data Object Descriptor) [64] to represent the measurement data as well as using the iRODS [78] data management system to archive the measurement data in a comprehensive, scalable and distributed way.

The measurement data can be represented as attribute/value pairs and the most important aspect of creating a structured representation of this data is to create the service advertisement to represent this measurement service capability, that can be queried and composed. Because there is an already ongoing work to represent the measurement data and service composition, we have not focused on stating a measurement specification in this thesis.

3.2.4 Potential Business Model

As we remarked before, our architecture must not only be adequate in defining the functionality that we aim to introduce, but also in striking the correct decomposition of that functionality between various different entities, such that these entities become natural and viable business models. This would enable businesses to emerge operating on one or another of our entity definitions, and be able to interact with other businesses using standardized interactions defined by the architecture. For a viable ecosystem to form, it is also necessary that each entity provide a sustainable business model, and possibly other conditions such as attainability and stability of a non-monopolistic model, in each business space. The richness of the ecosystem that can be potentially formed is another desirable metric.

A rigorous and complete consideration of these issues is beyond the competence of solely networking researchers, and the scope of this thesis. Our group is collaborating on an ongoing basis with the economics researchers to make progress on these issues. However, in this section, we include some brief discussion to provide some intuition regarding the viability of our architecture from this point of view, and also to show that the potential richness of our architecture is high.

For a viable business, there must exist a win-win situation with other business entities in the same economic domain, which implies that different players must have complementary strengths, so that each brings something to an interaction that the other values, but lacks the capability (or investment) to produce for themselves. Thus every architectural entity must indicate a stock-in-trade for a business operating in that role, and the functionality of the entity must produce added value. Every interaction in the architectural definition must realize an exchange of such added value between different entities; of course such an exchange may be mediated by some currency, or consideration. Finally, for the richness of the ecosystem, it should be possible for businesses to bring a variety of innovative value propositions to existing entities, not be restricted to just playing one of the few roles specified up front in the architecture.

In our verification architecture, we obviously consider the main entities we have proposed to have viable business cases. The stock-in- trade of the MSP is the measurement equipment at various Points-of-Presence in the Internet - an MSP with a larger variety of equipment at a larger number of exchange points in the Internet is in a position to contribute to the fulfillment of a larger set of verification requirements. The MASP provides a complementary value, that of algorithmic analysis of the collected data. The MASP's stock-in-trade is smart algorithms, and the value stems from the development of such custom algorithms, or the correct and efficient implementation of existing approaches. An MASP could use data-driven approaches using simulation or pattern detection over large volumes of pre-collected data as part of the unique value it provides.

The MCSP (which may be the same as ChoiceNet's general Composition Service Provider) functions as a sort of "market expert" of verification services, consisting of a knowledge base or expert system on what verification service offering exist in various marketplaces, and how they might be composed to best satisfy the verification needs of a particular chooser. The MCSP operates entirely on service offerings available from various marketplaces, but the offline data mining and computation that the MCSP undertakes on an ongoing basis creates the value of its service, which a typical chooser would not be able to undertake for an individual service request.

The normal service providers also have a role to play in the verifiability of the services they deliver. For one thing, they may act as MSPs for their own services themselves, offering their own measurements together with their services as an added value, to increase confidence in their service. Path service providers already have Points-of-Presence inside the network, and can offer claims of service or proofs of service measurements of quantities such as throughput, delay, jitter. Storage, content, or compute service providers can provide similar measurements at endpoints, or other measurements more specific to their service e.g. a storage service provider can provide storage and retrieval time measurements. Other types of measurements can be made by path or endpoint service providers that, for example, verify endpoint processing services such as encryption (by verifying that cleartext is not possible to obtain or reconstruct at an intermediate location in the network path).

Beyond these basic entities, there might be many business models for innovative or partial verification services to the chooser, or other entities in the architecture. We note that these services (and the measurement services in general) do not have to be perfect or infallible in order to provide reasonable or even significant benefit to the customer in terms of making informed choices; a fuzzy or approximate measurement may suffice the customer to inform their subsequent choices. Many existing measurement methods may have an active component, such that some packets (or fields in packets) are injected in the data stream to aid verification. Such injection needs the endpoint(s) to interpose operations in its local stack, or execute code in an existing stack protocol. In Section 4.3, we use such an example of injecting a field in the application PDU that allows some packets to be "marked". Providing such plug-in code can itself be a service, that is offered in conjunction with (or with reference to) a particular type of measurement; the chooser can then choose to buy the plug-in from one vendor, while also contracting for service with an MSP who offers the capability of making measurements based on the same measurement type, or plug-in.

Completely different mechanisms may co-exist, or evolve from or to the above ones; for example many customer-provider interaction may initially proceed without any verification (tantamount to trust, the current status quo), with later introduction of verification mechanisms. As another possibility, the service providers may themselves offer claims or proofs of service, and the measurement service provider may simply provide estimates of how dependable such claims are from spot checks (not necessarily related to this particular customers traffic) thus acting essentially as reputation brokers.

We have assumed above that the MCSP is able to translate the chooser's verification requirement into requirements for measurement. However, for chooser requirements that are very high level, abstract, or sophisticated, part of this translation may be a separate service itself. Such a Measurement Metric Converter Service might translate a a "VoIP" service requirement to a requirement to measure one-way delay, throughput and jitter as the quantities to be measured, and may further make the expert suggestion that satisfactory values for those should be < 80 ms, > 200 Kbps, and < 5 ms respectively. Such a provider could utilize algorithmic models of user experience as their stock-in-trade, or mine available past data on metric values and subsequently stated user satisfaction.

Considering privacy concerns, we can envision a capability enabling customers to pay and receive measurement service without exposing their identities. One possible solution could be a “proxy ID service” where only the proxy ID provider knows the chooser’s real identity and it provides a “temporary valid session id”, with some money in it, to the customer. Thus MSP essentially knows nothing about the identity of the customer, preserving the privacy at least partially.

Although measurement and verification services are likely to primarily rely on passive measurement (due to cost), they might partially reap the benefits of active measurement by having “track record” reports of such tests. Thus a “NSP reviewer service” can act as a regular customer running active measurements across NSPs. (Note that the proxy ID service can also help hide reviewer service hide its identity and thus NSPs differentiating reviews traffic.) Similarly, MSPs themselves can be subjected to “peer review” by other MSPs for quality assurance, or by NSPs for certification of non-interference and non-invasiveness of the measurement activity. Along the same lines, some customers might specifically want to make analysis of their past MD or of general aggregated MD. Therefore storage of past MD is desirable. A chooser can simply store such MD, or pay for a storage service to do so; or this capability can be bundled as part of MASP service by a larger provider. In the latter cases, the chooser may also then decide to make such measurement data archives available to other users, as a public service, or for a fee (becoming an NSP reviewer service provider).

Not all verification interactions need be mediated by money. Seattle [22, 160] is a community-driven educational testbed which consists of a large number of diverse computing devices (laptops, tablets, smart phones), each running a node manager “software” which ensures experimental applications installed on these nodes do not cause any security risk or performance problems while the owners use these devices. Experimental applications are written in a python-like language which utilizes the Seattle API providing both abstraction and security. Seattle is used for running simple end-to-end measurements of throughput, similar to Iperf, on whatever nodes are available running Seattle at a given time. The system is free, utilizing some cycles on devices that their owners contribute to the common cause. One can envisage a similar non-profit consortium of users, performing such experiments, building up a tomographic performance picture of parts of the planetary network over time.

The above discussion shows us that the architecture is capable of sustaining many different business models, thus increasing our confidence that some or many of them will be viable ones, and that the availability of the primary entities that support the basic architecture will be valuable enough to make them viable as well.

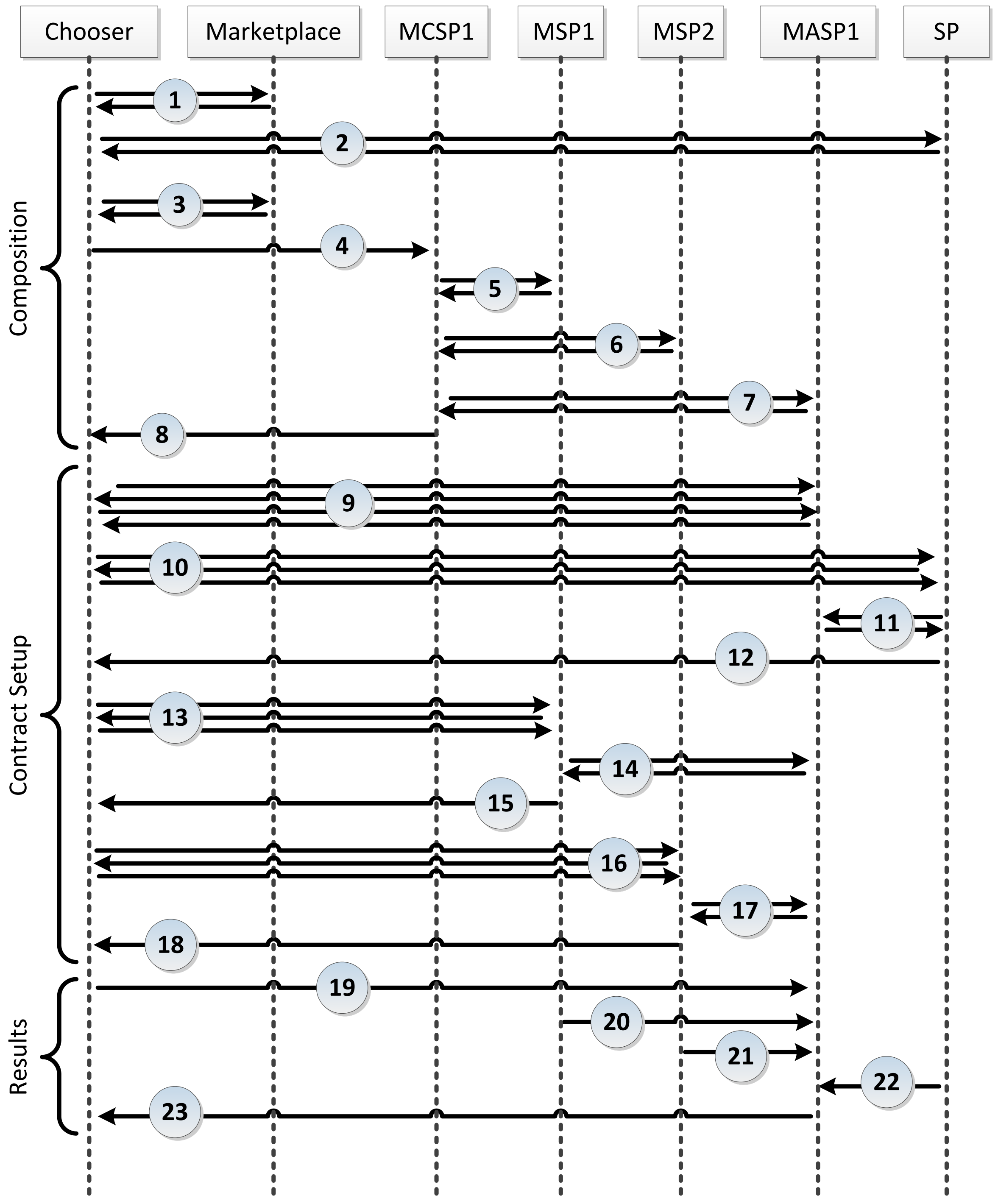


Figure 3.6: Time-Sequence Diagram for Interactions between entities. Each number M represents a message, which is referred as Message M in this thesis.

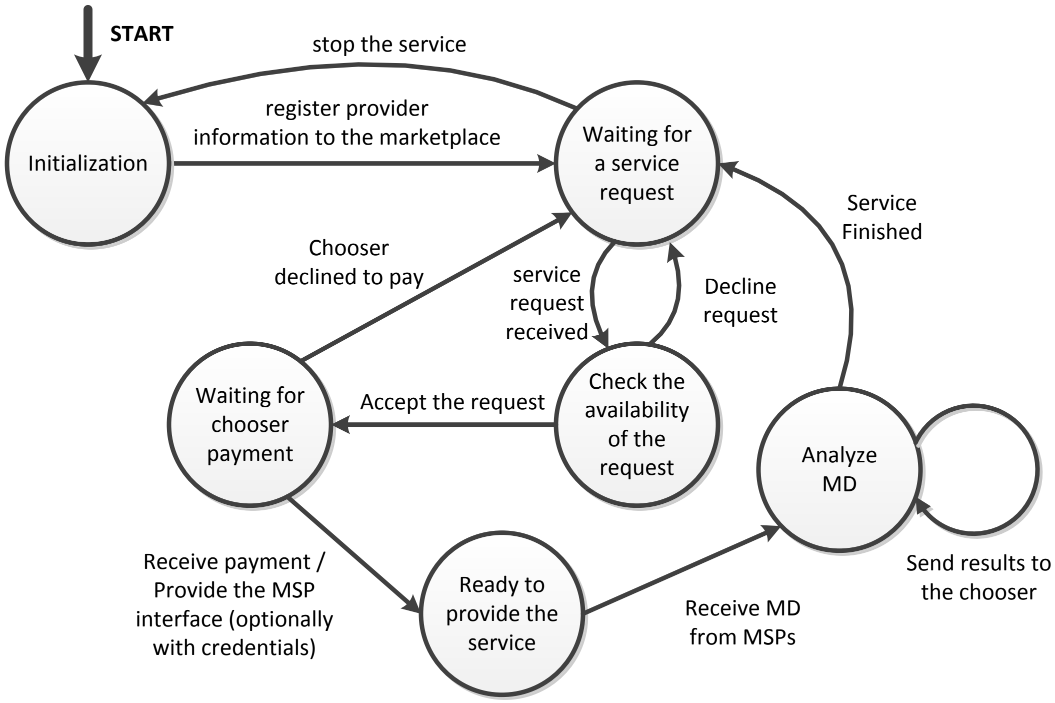

START
stop the service
register provider information to the marketplace
Initialization
Waiting for a service request
Service Finished
Chooser declined to pay
service request received
Decline request
Waiting for chooser payment
Accept the request
Check the availability of the request
Analyze MD
Receive payment / Provide the MSP interface (optionally with credentials)
Ready to provide the service
Receive MD from MSPs
Send results to the chooser

Figure 3.7: MASP State Diagram

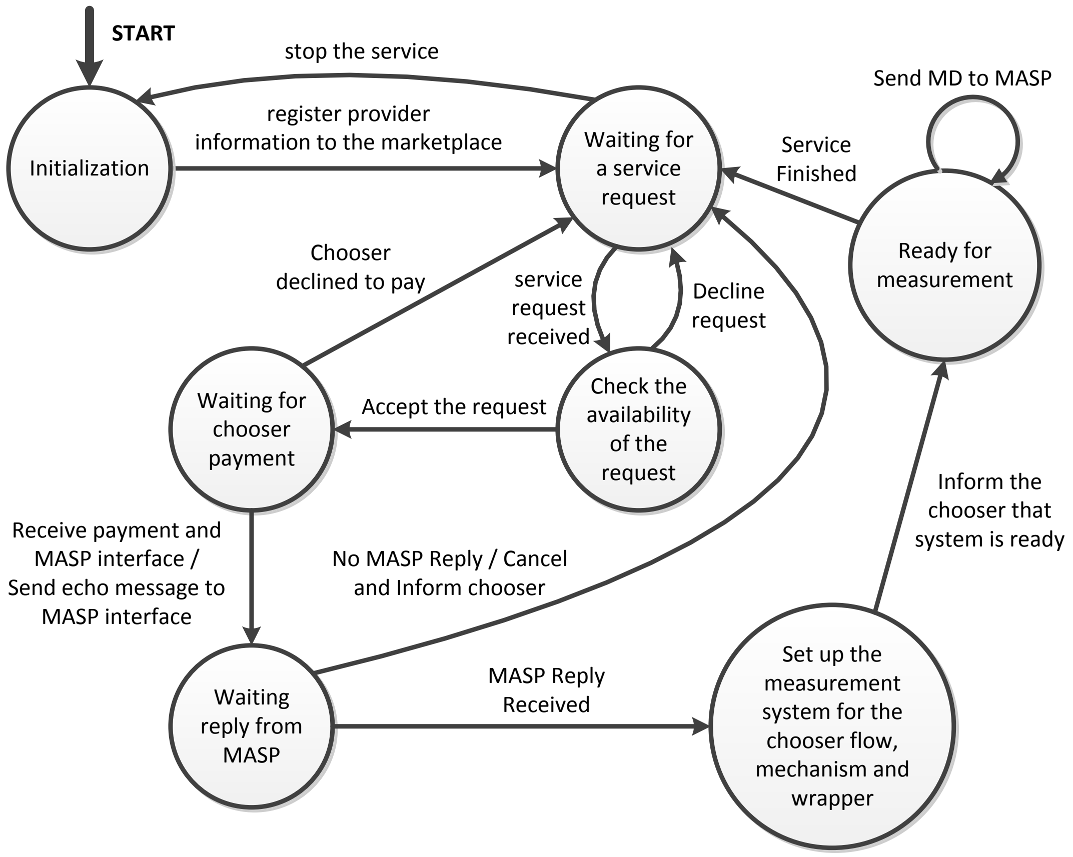

START
stop the service
Send MD to MASP
register provider information to the marketplace
Initialization
Waiting for a service request
Service Finished
Ready for measurement
Chooser declined to pay
service request received
Decline request
Waiting for chooser payment
Accept the request
Check the availability of the request
Inform the chooser that system is ready
Receive payment and MASP interface / Send echo message to MASP interface
No MASP Reply / Cancel and Inform chooser
Waiting reply from MASP
MASP Reply Received
Set up the measurement system for the chooser flow, mechanism and wrapper

Figure 3.8: MSP State Diagram

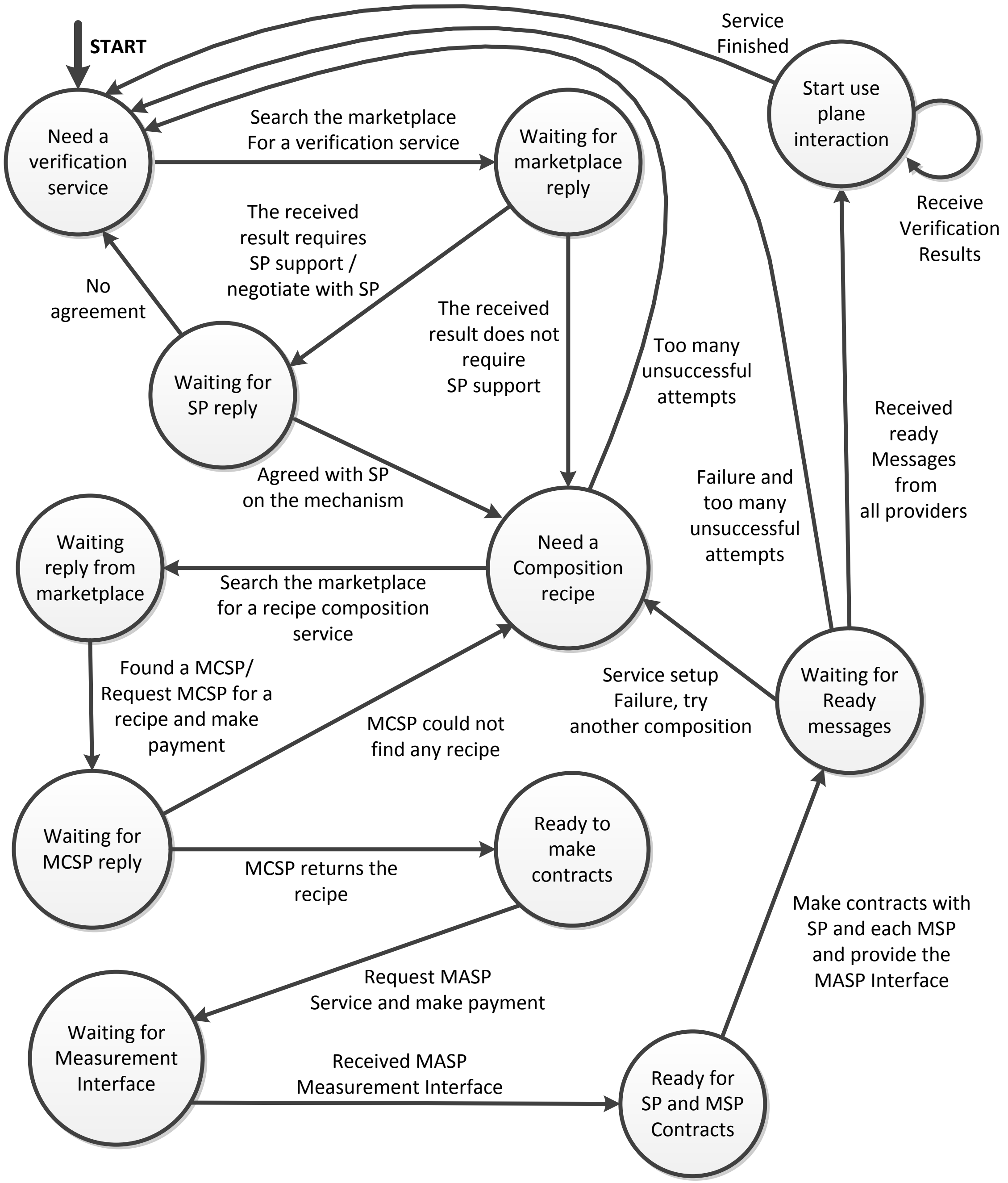

START
Need a verification service
Search the marketplace For a verification service
Waiting for marketplace reply
The received result requires SP support / negotiate with SP
No agreement
Waiting for SP reply
The received result does not require SP support
Agreed with SP on the mechanism
Too many unsuccessful attempts
Service Finished
Start use plane interaction
Receive Verification Results
Received ready Messages from all providers
Failure and too many unsuccessful attempts
Need a Composition recipe
Waiting reply from marketplace
Search the marketplace for a recipe composition service
Found a MCSP/ Request MCSP for a recipe and make payment
MCSP could not find any recipe
Service setup Failure, try another composition
Waiting for Ready messages
Waiting for MCSP reply
MCSP returns the recipe
Ready to make contracts
Request MASP Service and make payment
Make contracts with SP and each MSP and provide the MASP Interface
Waiting for Measurement Interface
Received MASP Measurement Interface
Ready for SP and MSP Contracts

Figure 3.9: Chooser State Diagram

Chapter 4

The Performance of a Verification Service

To show the value of our proposed framework discussed in Chapter 3, we designed several network measurement systems that fit into it, which we describe in this chapter. While some of them predates the ChoiceNet architecture, their goal is still the same and so all of these systems nicely fit in the picture of the ChoiceNet verification service architecture. In addition, to realize the "vote with your wallet" principle (referring to Section 3.1.2) of the ChoiceNet, we needed an agent that makes informed choices on behalf of the user and we utilized an earlier project of our research group, called SILO (explained in Appendix A.1). SILO's Tuning Agent can run various algorithms and provides automated decision making based on the feedback received from the network and we use this capability in Section 4.1 and 4.2. Note that, any other agent (including the user) could have been used to represent this decision making.

In each case, we aim for a meaningful system capability that is difficult if not impossible to achieve without the ecosystem of entities in our verification architecture, but becomes straightforward with the separation of roles and assumption of the interfaces we postulate. The first of these is a scenario in which measurements from the optical transport path is utilized by the end user's software agent to make choices between alternate path services, or the more fine-grained choice of the power level to use. In the second case, we construct a similar scenario, but with a wireless network, integrating pre-existing open source software modules to play the roles of both the measurement provider, and the user's agent that receives measurement data to make choices. In our third scenario, we create a simulation in which each role is played by simulation modules, to showcase how the separation of roles allows both simple and more sophisticated analysis based on the same measurement data. Having promising results in the simulation, in the fourth case, we give the results of an actual prototype deployed on a meso-scale testbed called GENI, explained in Appendix B.1. We also briefly discuss the implementation details of a minimum and yet complete ChoiceNet prototype with Verification Service in Section 4.5.

4.1 Verification based Path Decision in Optical Transport Service

The first part of our case studies is about the system we built on GENI, a national, virtual testbed to create custom network topologies (for a detailed explanation of GENI, readers are referred to Appendix B.1). As part of the GENI Instrumentation & Measurement Group, IMF (Integrated Measurement Framework) [75, 76] is a collaborative research project between North Carolina State University, Columbia University and University of North Carolina at Chapel Hill/RENCI, which utilizes optical physical layer characteristics with cross-layer capable platforms and then provides these capabilities through interfaces of other measurement frameworks such as PerfSONAR. This is essential for GENI, because GENI provides virtual machines to experimenters, and in such a virtualized environment, it is typically not possible for experimenters to directly access to the real physical layer, which is essential to be able to observe and leverage the optical link characteristics by their cross-layer protocols. However, note that in this section, we only give the details related to our verification architecture, and more information about IMF can be found in [75, 76] for interested readers.

As shown in Figure 4.1, our case scenario [184, 185] has a GENI slice consisting of two virtual machines (VM) that communicate over the optical fiber provided by BEN [131]. Both VMs are running SILO (discussed in Appendix A.1) stacks as the "end-user software agent" which makes cross-layer decisions. The VM on the left is streaming video to the VM on the right through the optical channel, on which the optical switches periodically publish measurement data to an XMPP [191] server, in terms of optical power and bit error rate (BER), where XMPP (eXtensible Messaging and Presence Protocol) is a rendezvous protocol, enabling XMPP clients to login and find each other by connecting to an XMPP server. Also, an additional important technology is XMPP PubSub (Publish and Subscriber) [154, 192] where some XMPP clients publish information to specific "topic"s and other interested XMPP clients can then consume this information by subscribing to these topics, without publishers and subscribers requiring to know each other as long as they have the credentials to publish or subscribe. The SILO stack on the left VM has two additional services; an XMPP subscriber service whose sole purpose is to subscribe to the topic about the optical measurement data, and a SOA (Semiconductor Optical Amplifier) controller service which sends XMLRPC requests to the NetFPGA to modify the optical power through SOA. There is also a special SILO Tuning Algorithm running on the left VM for this experiment, which periodically queries the SILO service and issues commands in real-time to the SOA controller service to increase the power level to achieve better video quality if BER increases. Moreover, if the Tuning Algorithm decides that it can not compensate anymore by increasing the power level, it switches the path by using an alternative link (Eth1) for streaming. The results [184, 185] show that the video quality is significantly increased by this measurement and actuation feedback loop.

From our verification architecture point of view, it is fairly easy to see how each entity fits. The customer is the video streaming application (on the left VM) choosing the SILO stack and then this verification service provider. The measurements are performed by the optical switches acting as the measurement points (MPs) and the XMPP server combined with MPs is the Measurement Service Provider (MSP) which provides the measurement data (MD). The analysis component (MASP) is the Tuning Algorithm which decides whether the optical path service is providing the expected quality of service. Additionally, the Tuning Algorithm acts on behalf of the customer, by choosing another path as in the "vote with your wallet" principle of ChoiceNet (discussed in Section 3.1.2).

Another goal of our unified architecture is to use existing measurement frameworks (discussed in Section 2.2.2) to enable a commonly accepted software mechanism between MSP and MASP (described in Section 3.1.2) so we (i) added PerfSONAR Measurement Point capability to the IMF measurement handler of the switches as the MP, (ii) created a database as the measurement service provider [75], and (iii) modified the PerfSONAR UI [85] to visualize both archived and the real-time optical MD. Figure 4.2 is a screenshot of the modified PerfSONAR UI, displaying the power-level measured in two weeks. This work shows that we do not advocate discarding existing measurement platforms but rather enabling them to coexist in the same unified service ecosystem.

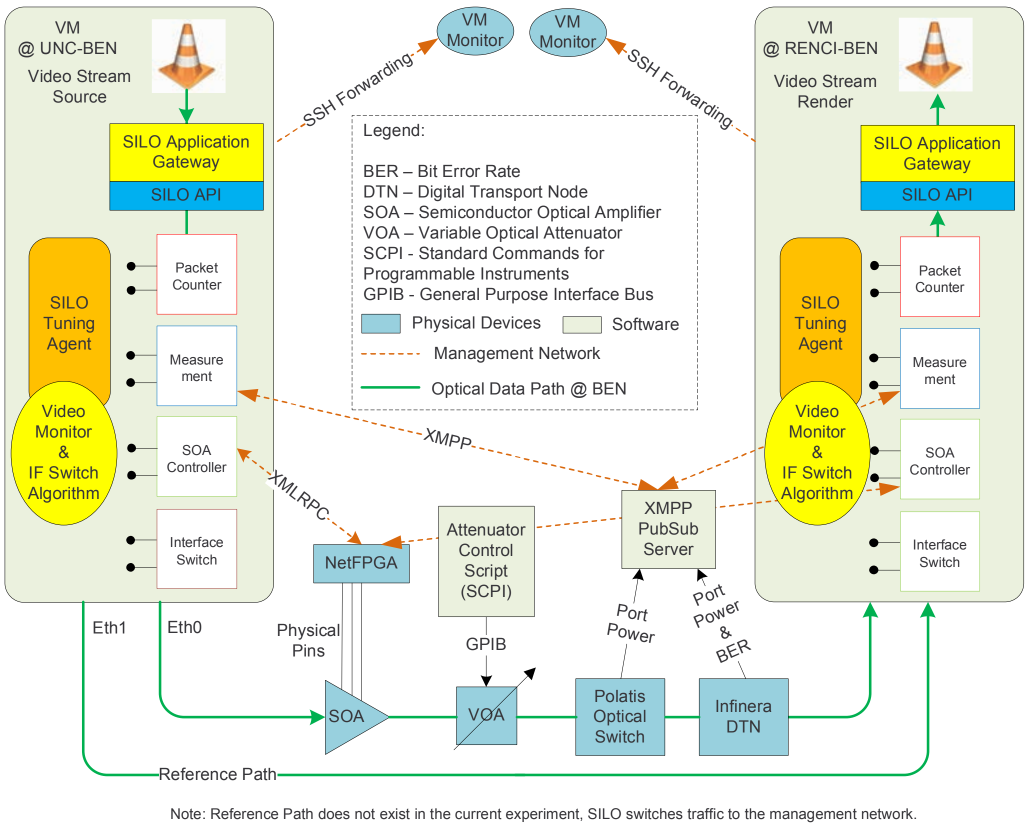
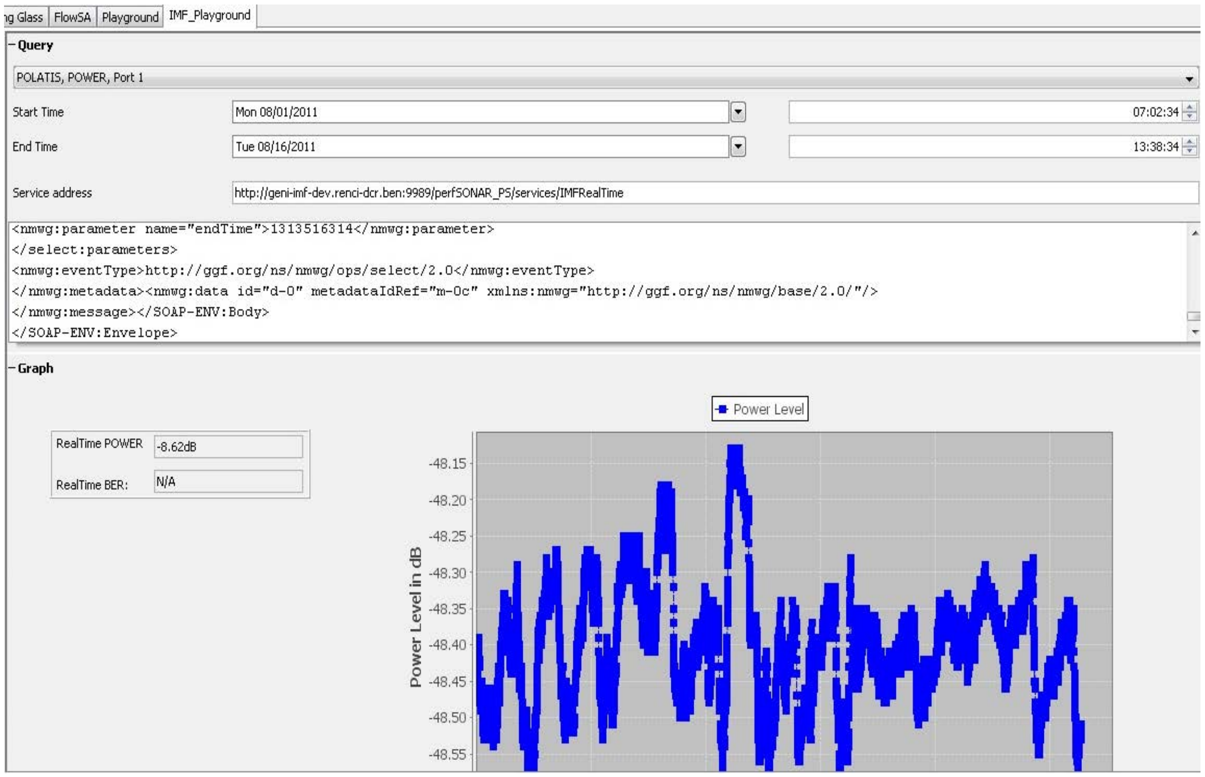


Figure 4.1: GEC8 IMF Demo (figure taken from [184])

4.2 Wireless physical layer Alternative using Standard Choice Interface

Our second case study presents a wireless network framework that our architecture fits into. We first give a brief information about the outdoor testbed we ran our tests on, called CentMesh, followed by an overview of OMF, a well-recognized control and management framework focusing on repeatability. Finally, by utilizing OMF as the MSP and SILO as the software agent for automated decision maker on behalf of the user, we show a real cross-layer closed-loop control scenario and present some experimental data. While the data or the tuning algorithm themselves are not very important, they show that the architectural approach is valid.



POLATIS, POWER, Port 1

Start Time Mon 08/01/2011 07:02:34

End Time Tue 08/16/2011 13:38:34

Service address http://geni-imf-dev.renci-dcr.ben:9989/perfSONAR_PS/services/IMFRealTime

```
<nmwg:parameter name="endTime">1313516314</nmwg:parameter>
</select:parameters>
<nmwg:eventType>http://ggf.org/ns/nmwg/ops/select/2.0</nmwg:eventType>
</nmwg:metadata><nmwg:data id="d-0" metadataIdRef="m-0c" xmlns:nmwg="http://ggf.org/ns/nmwg/base/2.0/"/>
</nmwg:message></SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

RealTime POWER -8.62dB

RealTime BER: N/A

Figure 4.2: PerfSONAR UI displaying results of the optical power levels for given time interval

CentMesh Overview

The CentMesh (Centennial Mesh project) [25] is a wireless mesh network testbed enabling researchers to run and test their innovative protocols in a real outdoor environment located in Centennial Campus of North Carolina State University. Its software is open-source and its wireless channels are based on 802.11a/b/g by utilizing the MadWifi Driver [104]. It has stationary and mobile nodes as pushcarts. In this section, we only use the pushcarts in our experiments. Its high-programmability allowed us to install our applications and also modify the network interfaces to easily run our experiment as we describe in Section 4.2.

OMF Overview

OMF (cOntrol and Management Framework) [146, 126] is a control and management framework for testbeds to easily setup the testbed nodes and run experiments. OMF has a special emphasis on the repeatability of the experiments, since repeatability of an experiment adds much more value to that research. To achieve this goal, all steps of the experiment setup and execution are defined in an “experiment description” (ED) script written in a high-level language and given to the OMF as input, so that the same ED can be used by other researchers to produce similar outputs in similar conditions.

As shown in Figure 4.3, OMF consists three main components; (1) the experiment controller (EC) which is the user interface component enabling interactions between the experimenter and the OMF system, (2) the aggregate manager (AM), the manager component keeping testbed and node information (such as IP and MAC addresses) to coordinate the setup of the experiment, and (3) the resource controller(s) (RC) is the component installed on all OMF testbed nodes to receive requests from OMF system in order to run experiments and send responses.

Once a user issues a command to start the experiment, EC reads the configuration in ED and queries AM with a HTTP request/reply (OMF version 5.2) to find out available nodes. AM replies with the IP addresses of nodes and once EC gets this reply, all further communication between EC and RC is entirely based on XMPP PubSub (discussed briefly in Section 4.1). Additionally, in order to minimize traffic interference, OMF suggests an isolated network for control & management, other than the experimental network to run experiments. In our experiments, we used a wired network for control and the wireless mesh network (CentMesh) to run our experiments.

OMF Experiment

In this case study, the MSP is the OMF system and the user’s agent is the SILO stack that receives the MD, analyzes the results as MASP and then makes choices accordingly; for example changing a node’s wireless transmission power level to achieve the desired quality of service. In order to realize this, we first studied the latency issues involved. While the obvious way for SILO to communicate with OMF is to let SILO be the experimenter and create new experiment each time SILO Tuning Algorithm wants to change some physical substrate parameter, that will create a new “experiment” per actuation causing as much as 16 seconds. Such a timescale rules out many cross-layer adaptive experiments and so we decided to have one long standing experiment description (ED) controlled by an “outside” adaptive algorithm that we call the “slave” method, facilitating efficient control but reducing the repeatability of experiments. Also our methodology was to capture the

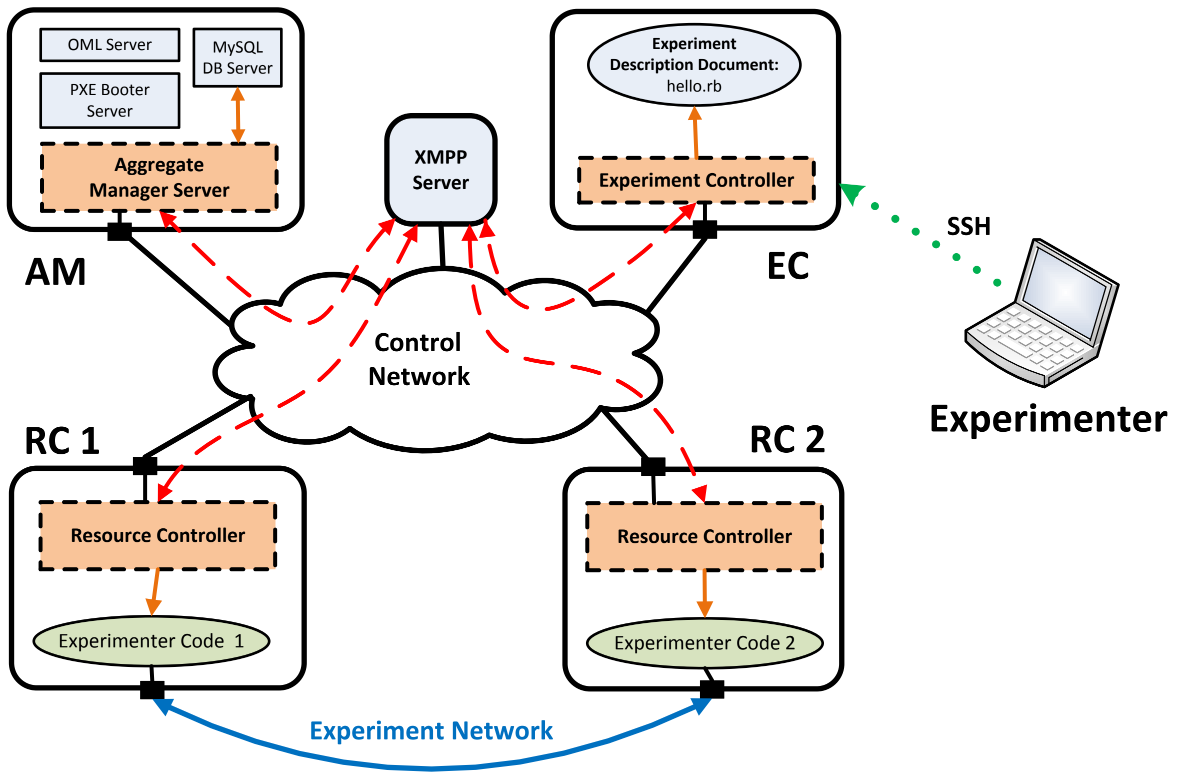


Figure 4.3: OMF architecture. Dashed orange boxes represent core OMF entities. Also, the dashed red arrows represent the interactions, that are all going through the XMPP server, whereas the solid blue arrow represents the experiment network.

final substrate actuation command at the point of actuation, so we short-circuited the OMF return path and also did not use OMF for measurement.

As a validation of our proposed *"slave"* method, we carried out a real experiment of the topology shown in Figure 4.4, where two push-carts have desktops as testbed nodes with batteries and attached wireless antennas. The carts also have an Ethernet interface to create the OMF "control network" and each testbed node runs Fedora 9 with OMF RC installed on them. Additionally, one pushcart has SILO prototype installed on Fedora 9 and one laptop running OMF version 5.2 AM and EC on Ubuntu 10.04 to set up the experiment. The experiment involves one node to continously send an Iperf UDP stream to the other over the wireless medium for varying distances. Moreover, the sender and receiver testbed nodes periodically send wireless transmission power and received throughput respectively as MD to the SILO stack through TCP sockets. Our

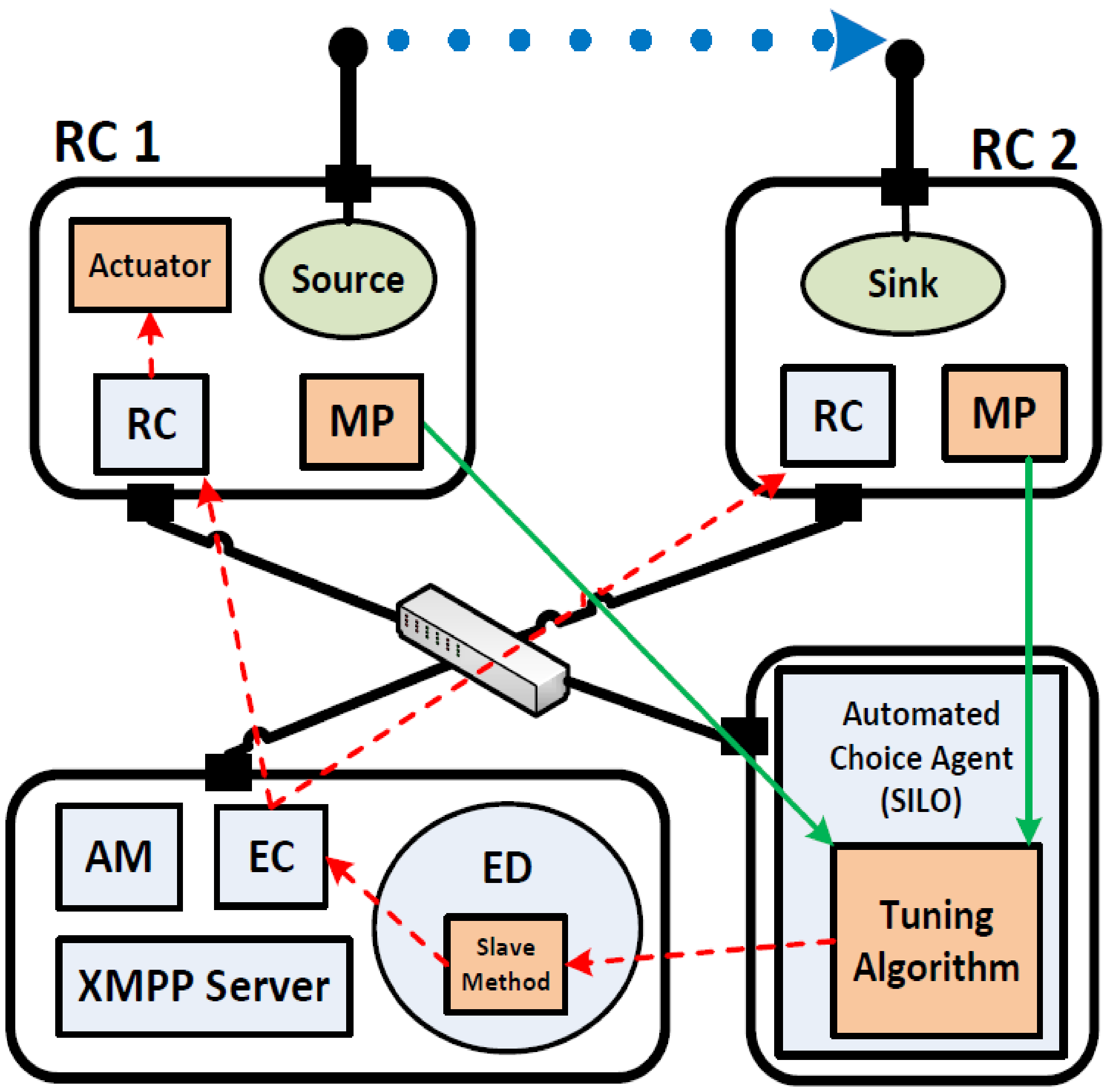


Figure 4.4: OMF as a Service Provider. All rounded rectangles represent computers connected together through a regular switch to form the wired OMF Control Network, where orange rectangles denote the core components of this experiment. In addition, the red dashed arrows represent this control interactions, whereas the green solid thin arrows denote the measurement data transfer. Also, the blue thick dotted arrow symbolizes the experiment's data transfer over the wireless channel.

SILO Tuning Algorithm then analyzes this MD and then "adjusts" transmission power accordingly. This is similar to the scenario in Section 4.1 because the OMF testbed nodes are acting as MSPs, and the software agent (SILO stack and Tuning Algorithm) has the MASP role. Furthermore, the SILO Tuning Algorithm acts on behalf of the customer, because it "chooses" to adjust the sender node's transmission power during various conditions, such as the varying distance between the sender and the receiver.

Our goal is not to come up with a "great" tuning algorithm but to demonstrate this capability with a simple tuning algorithm that tries to keep the received throughput between 12Mbps and 18Mbps. That is, if the throughput is higher than 18Mbps, the algorithm decreases the transmission power for power saving, whereas if the throughput is lower than 12Mbps, it increases the transmission power. Our wireless cards with Atheros chipsets support open source Madwifi [104], and provide 1 to 17 dbM transmission power. The maximum throughput observed with these cards is 29Mbps.

Table 4.1: Outdoor Experiment Results Table

Distance(feet)	Power Ratio(dBm)
60'	17 (still 7Mbps)
50'	17
40'	12
30'	8
20'	4
10'	1
5'	1 (25Mbps)

The results in Table 4.1 show a high correlation between the distance and the transmission power. While the distance up to 10' show almost the same for any power level, the effects of distance is clear around 60' that SILO tries to use maximum power to get the highest possible throughput. We emphasize that the data or the tuning algorithm itself are not very important but the experiment presents yet another case study showing that the Choicenet architecture fits into both as a verification service and also as a actuation (vote with your wallet) case.

4.3 Decoupling Measurement and Analysis

In this case study, we realized a prototype in NS-3 for our verification architecture, to first show a simple and then complicated experiment, where the entities are decoupled and are collaborating through the ChoiceNet interfaces. The purpose is not to design an optimal system in any sense, but to show that even a simple and straightforward mechanism can provide meaningful feedback to the user, simply by allowing the user to form a partnership with the measurement service provider, and access information within the network cloud. In the simulation, we use "Client" to represent a chooser who also incorporates its own MASP, and "Server" to indicate the end service provider (SP in Section 3.2). We first introduce our novel yet simple mechanism, called the "marker mechanism" with a discussion on its benefits, followed by the experiment topology and the comparison of its results with existing measurement tools. Next, we give a more complicated simulation that apportions the overall jitter contribution of each provider in a video streaming case, in order to show how the power of separation of roles achieves a harder verification task.

Simple Throughput Measurement

As the name implies, the "marker" mechanism is based on marking some of the chooser's packets, while the chooser is sending a stream of data traffic to the destination. Note that this is a case where the chooser runs a plug-in agent of the measurement service provider, as mentioned Section in 3.2. The pseudocode of the mechanism is shown in Algorithm 1, where each marked packet has a sequence number to distinguish that marker packet and in our simulations, the sequence number is stored at the application payload of UDP packets. When a measurement point (including chooser and SP) captures the initial marker packet (say sequence number 1), it records the timestamp as t_1 and start adding packet lengths of packets belong to that flow, received after this initial marker packet. When the chooser sends the next marker packet at time t_c, each MP (including chooser and SP) records the timestamp of this new packet as t_2, calculates the number of bytes recorded, called B between t_1 and t_2, and then sends an MD to MASP with five tuples; the sequence number, t_1, t_2, B and MP ID. MASP (the analyzer) can then calculate the throughput as $(\mathsf{B}/(\mathsf{t}_1 - \mathsf{t}_2))$ and OWD (one way delay) as $(\mathsf{t}_2 - \mathsf{t}_c)$ for each MP. However, it is up to MASP to do any additional calculation to perform better analysis.

Algorithm 1 MARKER(flow f)

```
while ($p$ = recvFromFlow($f$)) do
  if $p$.isMarker() then
    if $p$.isFirstMarker() then
      $f[cTime] \leftarrow$ GetTimeNow()
      $f[lastSeqNum] \leftarrow 0$
      $f[rxBytes] \leftarrow 0$
    else if $f[lastSeqNum] < p$.seqNum() then
      $f[pTime] \leftarrow f[cTime]$
      $f[cTime] \leftarrow$ GetTimeNow()
      $f[rxBytes] \leftarrow f[rxBytes] + p$.packetSize()
      send(mp_id, $p$.seqNum(), $f[rxBytes]$, $f[cTime]$, $f[pTime]$)
      $f[rxBytes] \leftarrow 0$
      $f[lastSeqNum] \leftarrow p$.seqNum()
    else
      $f[rxBytes] \leftarrow f[rxBytes] + p$.packetSize()
    end if
  else
    $f[rxBytes] \leftarrow f[rxBytes] + p$.packetSize()
  end if
end while
```

Discussion about the marker mechanism: Under ideal conditions, i.e., perfect time synchronization, no packet loss, no queuing delay and no out-of-order packets, the marker mechanism should give 100% accurate throughput because it will calculate the same exact number of bytes at each MP between t_2 and t_1. Also, perfect synchronization naturally gives the perfect one-way-delay at each MP. We observed this expected phenomenon in our simulations. We also note that losing a regular (non-marker) data packet does not affect the accuracy of the mechanism, because MPs are still able to correctly count the number of bytes received between marker packets, and in a typical 1% packet loss scenario, losing a marker packet is relatively rare. Thus, the effect of low packet loss rate on the overall system is also low. However, marker packets can get lost and in such cases, an MP will keep counting the number of bytes until the next marker packet arrives. When it receives the next marker packet, the MP will then send MD as shown in Algorithm 1. Now the analysis (MASP) provider can interpret the loss of a marker packet by the sequence number and can average the throughput accordingly. The same holds also for multiple marker packet losses. Similarly, out-of-order regular data packets do not affect the accuracy of measurement and in some cases, where the marker packets themselves arrive out-of-order, MP simply discards the smaller one by keeping a counter of the maximum sequence number received as shown in Algorithm 1. Variation in queueing delay affects the throughput because it causes some packets to arrive later than normal (reducing throughput) and others to arrive quicker and causing peaks in throughput. However, its effect is relatively low especially in low granularity high throughput measurements. Combined with one way delay measurement, MD of marker mechanism can reveal which provider is causing the queueing delay as we discuss in the simple experiment.

Flexibility: The marker packet mechanism is chosen by the chooser, but plug-in is installed by the SP. We have already shown this possibility in Section 3.2. One of the benefits of such a mechanism is its flexibility, i.e., marker packets can be sent at any time the chooser prefers, and therefore it can be increased or decreased depending on the throughput or granularity level. Note that, it does not have to have a specific time or packet size, and marker packet information (such as the sequence number) can be inserted inside a data packet as well. In the following paragraph, we describe how such a mechanism can be realized in practice.

Marker Packet Identification: The "marker mechanism" require identifying each marked packet uniquely. Even though a unidirectional flow is identified by the five tuple (IP src/dst, transport src/dst port and protocol), it is harder to uniquely distinguish a flow's packets. In theory, the IP Identification field [186] can be employed however, in practice, it is not a reliable way as discussed in Section 2.2.1. One way is to add a sequence number to application field at UDP packets, which is incremented at every marker packet. For TCP, sequence numbers can provide partial uniqueness; while retransmissions and duplicate packets are possible, MPs can store the maximum sequence number of a marker packet encountered, and can then accept a marker packet only with a larger sequence number. This avoids retransmission and duplication related problems at MP and can provide some degree of uniqueness, without changing socket implementations. In such a case, the marker packets can be identified by taking the modulo of the TCP sequence number, where modulo number is exchanged in the measurement setup. We now describe the simulation setup and then show its results.

Simulation Setup: Figure 4.5 shows our topology in NS-3 simulator [125], where the SP (the end service provider as discussed in Section 3.2) sends variable-sized UDP packets with varying intervals to the chooser through NSP1, MP1, NSP2, MP2 and NSP3 nodes. A close approximation can be done by eliminating that MP (plug-in), assuming that the SP sends out frames regularly, and any jitter observed by the first MP is the fault of the first NSP. MP1 and MP2 are never bottleneck links and they are the measurement points capturing all of the packets on the way, running the measurement mechanism and sending MDs to MSP1. MSP1 then sends the MD to the chooser. MASP is also located at the chooser and receives MD from the chooser, MSP1 and SP. BG1 and BG2 send TCP and UDP background traffic to BG3 and BG4 through SP1, and BG4 also sends traffic to the chooser through NSP2 and NSP3. In this scenario, we ensure that MP1, MP2 add no additional delay and their outgoing links never get congested.

Simulation Results: We now show the result for throughput and one way delay in Figure 4.6 and Figure 4.7 respectively. Our network has 1% packet loss for the path and 10ms queueing delay per packet on the average. It is important to emphasize here that the SP's sending throughput rate is varying (not constant bit rate) with time and that is why, this throughput rate is also an important input parameter for throughput verification (unlike all existing measurement tools discussed in Section 2.2.4) which the

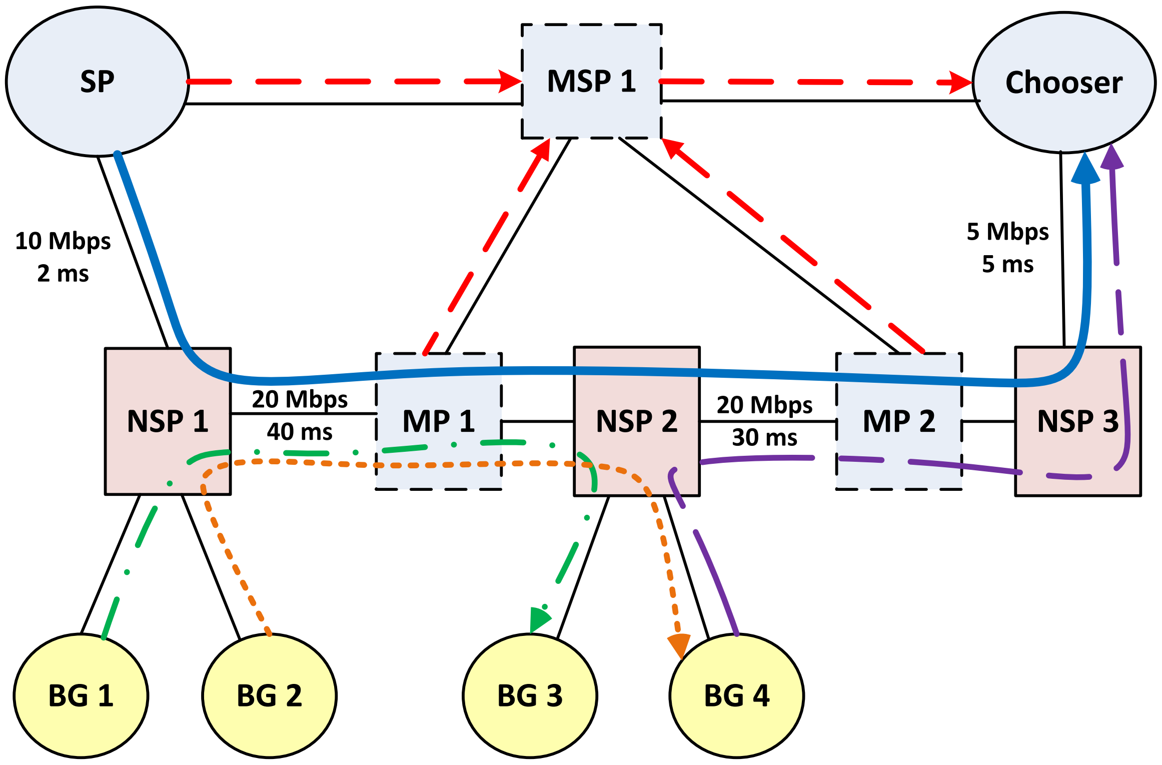


Figure 4.5: NS-3 Topology. In the figure, the dashed blue boxes indicate the MSP components, whereas the solid pink boxes indicate the NSPs (network service providers) with their bottleneck links by the unidirectional bandwidth and delay values. Note that, all other links have much higher bandwidth and much lower delay. In addition, the oval yellow nodes represent the background traffic nodes, whereas the oval blue nodes represent the SP and the customer. The dashed green, orange and purple arrows show the background traffic, whereas the solid thick arrow shows the SP flow. Finally, the red dashed arrows represent the MD traffic.

makes the problem more challenging. As shown in Figure 4.6, even a mechanism as simple as the marker mechanism achieves a high accuracy. We also note the two interesting observations; i) in Figure 4.6 at $t = 4.7$, the throughput measured at MP1 is significantly lower than usual and also at the next marker packet $t = 6$, the MP1 throughput is even higher than the SP's sending rate. This is due to the number of temporarily queued packets at NSP1 before $t = 4.7$, so all those waiting packets which are later sent by NSP1 are counted as part of the next measurement, therefore we can understand that queueing took place within that time interval. ii) in Figure 4.7 (not belonging to the same measurement) we can observe a peak in MP2 delay t=4, due to queueing delay of NSP2. However there is no change at MP1 measurements, therefore NSP1 is not to blame in that time interval, whereas at t=20, we have introduced an additional background traffic to NSP1 from BG1 and BG2, and that is why, the SP flow is affected significantly, whereas the contribution of NSP2 is low, therefore NSP1 is to blame within that time interval. We can also observe that NSP3 has not introduced any signicant performance problem during this simulation.

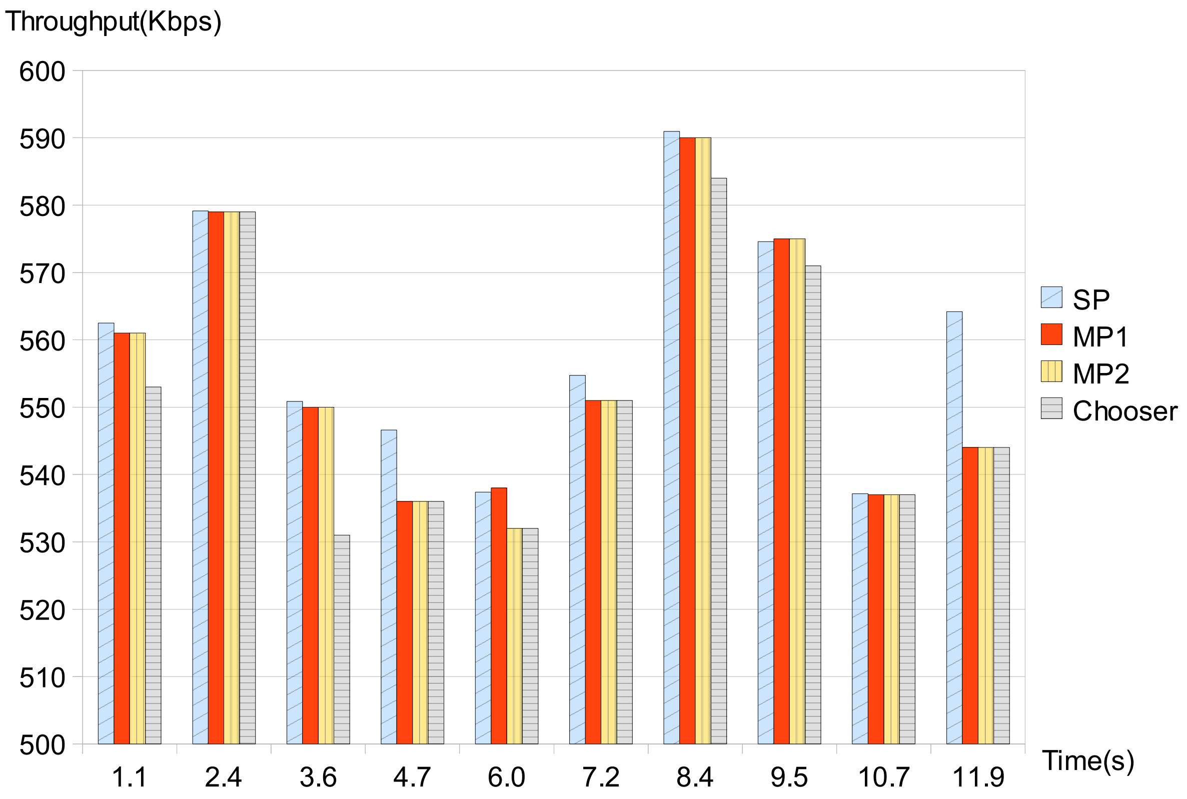


Figure 4.6: Throughput Results vs. Time for each entity, where each cluster represents a marker packet based measurement at each entity, and the time axis denotes the time SP sent this marker packet.

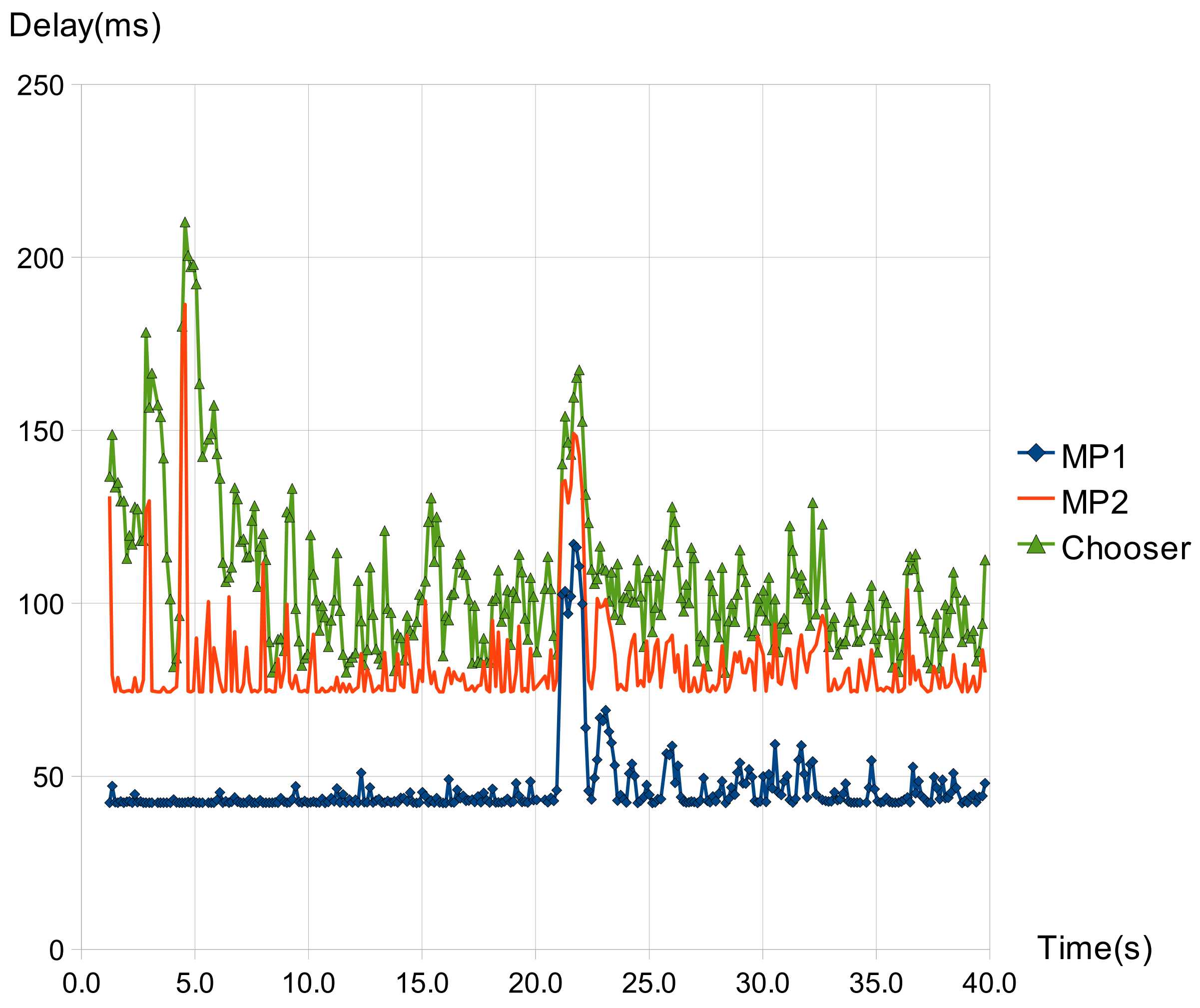


Figure 4.7: One-way-delay vs. Time Results for each entity (except client), where each data point represents a marker packet based measurement at each entity, and the time axis denotes the time SP sent this marker packet.

Overhead and Delay Benefits

As stated in Section 3.2, MD transmission should introduce low overhead on the network. It might appear MD being collected all over the network and being sent back to an analysis provider (MASP), which further sends the analyzed report back to the chooser, may impose severely increased overhead and delay. We provide the arguments to show that this is not necessarily true, and in fact the overhead may be far lower with the proposed architecture. Assume a scenario where each network service provider (NSP) has 3 hops. Then a path consisting of M=5 NSPs makes N=15 hops. Let the SP send data at 2Mbps. The pathchar [42] approach uses 64 packet probes for each of 45 different packet sizes (from 120 to 1528 bytes) in order to get a slope of available bandwidth. This makes 2880 packets and 2MB to send and ICMP error replies can add roughly 0.2MB load on the work. Also, the time it takes to send 2MB at 2Mbps is at least 8 seconds. However, in some cases, the accuracy we gain is less than 50%. An earlier comparison [95] shows that among three other measurement tools, the minimum number of packets sent is around 1000 and results still deviates more than 50% in some cases. Other end-to-end methods such as Pathload [81] claim 10-20% accuracy with again considerable network consumption. Further discussion is provided in Section 2.2.4. In the verification ecosystem approach above, there are $M + 1$=6 MD producers (including the chooser and SP3). At 2Mbps sending rate for a SP flow, a marker rate 1 in 100 with 512 bytes packets makes 5 marker packets on average. The marker information can be added to a regular data packet but conservatively speaking, independent marker packets (with 200 bytes data) add ~10Kbps extra network load. Also each role sends an MD upon receiving a marker packet so 5 marker packets at 6 MD producers make only 30 MD packets, compared to 1000s of packets in previous approaches. A conservative calculation makes 60Kbps total extra network load and it is possible to dynamically adjust the sending rate of marker packets.

In addition, one-way delay measurement is challenging due to the time synchronization issues and existing measurement systems such as Surveyor (discussed in Section 2.2.2) measure the one-way delay between the measurement nodes only, whereas our approach measures the delay of the actual customer flow packets, which provides a more relevant result to the customer's quality of experience. Furthermore, none of the existing measurement approaches (discussed in Chapter 2) reliably and specifically relate the measured performance to the issue of which provider is to blame when a sender has an unsatisfying experience. For example, if the sender makes a contract for a 2Mbps throughput, but instead sends at rates varying from 1Mbps to 1.5Mbps, then the observed throughput needs to be compared with the sending rate.

Overall, creating a verification ecosystem with measurement service providers can provide more robust and informative feedback to users about individual provider performances, at lower overhead and delay. In the following subsection, we describe a more complicated simulation to illustrate a harder verification task.

Jitter Apportionment for Video Streaming Experience Quantification

In this second part of this case study, we show a significantly sophisticated simulation by using the same exact MSP as the previous experiment, but a more complex analysis service (MASP) in a video streaming scenario. Such a new MASP service can be nothing but potentially an innovative new player entering the market, and all it needs to do is to have an algorithm that collects MD from MSPs and provides results to the chooser. Video streaming is commonly used today but the video experience of a user can vary greatly. While the user watching the video can easily tell the quality of experience, there is no easy way to 1) quantify the lack of satisfactory experience and more importantly 2) apportion the blame for unsatisfactory experience among multiple bandwidth providers.

We take jitter as a commonly accepted metric for video experience and focus on the second challenge. Using the MPEG-4 video traces available from past studies [181, 55], we have extended an existing publicly available NS-3 video streaming sender and receiver code [180] to show the effects of each provider on the end-user experience quantitatively, i.e., by counting the number of video player "freezes" due to buffer underflow during video play-out, and additionally the number of "glitches" which denote the frames or parts of the frames lost due to congestion. Our sender SP node is based on NS-3's built in ns3::UdpTraceClientHelper Class which sends the video frames at indicated times in the video trace file, but we have extended its capability in order to send larger video traces by fragmenting the video frames at the application layer and reassembling them at the application running on the receiver (chooser) side. By doing this, we easily send ~2Mbps throughput on the average, instead of the earlier ~200Kbps unfragmented video frames. We believe this modification gives a more realistic scenario because today, online video streaming providers such as NetFlix offer streaming rates ~5Mbps [124]. In addition, the artificial video player at the receiver, requires at least 25 frames in its buffer to play the "video". If there are not 25 frames in the buffer, then the video player "freezes", waits for the buffer to fill up and retries to "play" at the next second. Additionally, if a frame is lost or a part of the frame is lost, the frame is discarded and is not retransmitted. Note that in this experiment, there is no packet loss due to physical medium characteristics, i.e. all losses are due to queue buffer overflow.

The experiment topology is the same as in Figure 4.5 and we are also using the

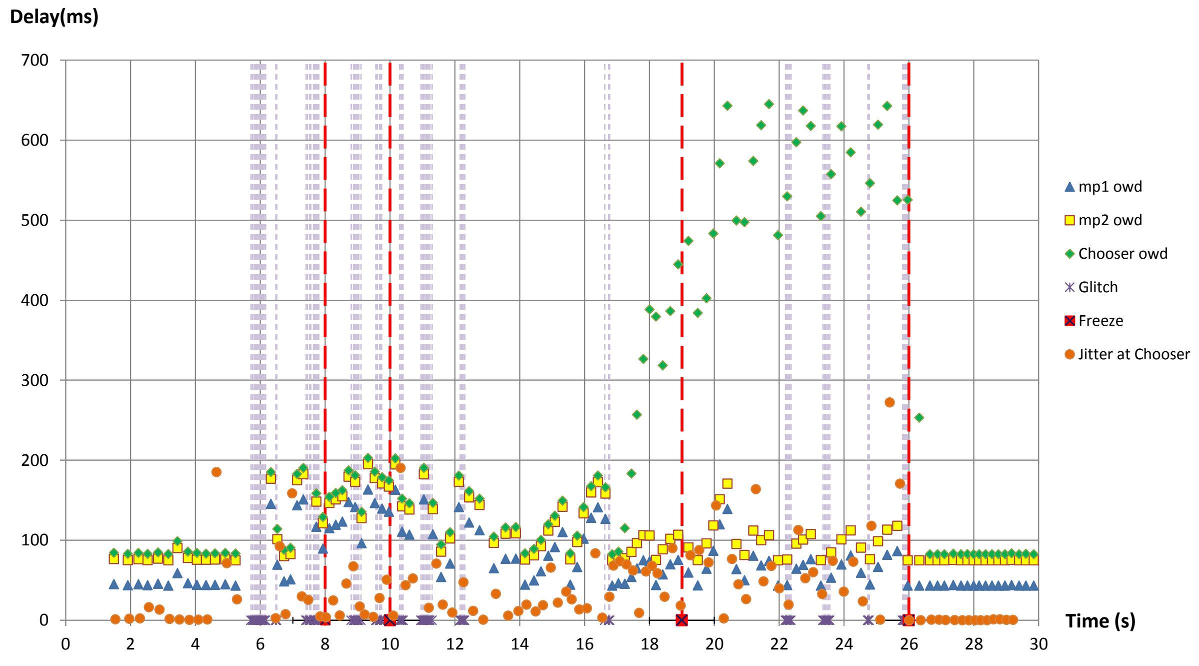


Figure 4.8: OWD, Jitter and User Experience Quantification, where each owd data point represents a marker packet based measurement at each entity, and the time axis denotes the time SP sent this marker packet. Also the vertical lines denote the times a freeze or glitch happen with their respective icons.

marker mechanism in this experiment. However, the traffic pattern is different. Figure 4.8 shows the one-way-delays measured for each marker packet at MP1 (measurement point 1), MP2 and the Chooser respectively. The figure also shows the jitter (packet delay variation) experienced at the chooser by taking the OWD difference between a marker packet with sequence number i and a previously received marker packet with sequence number $i-1$, as described in [40]. While these are important metrics, our goal in this experiment is to quantify the user's experience, which in this case is the number of freezes and the glitches that reduce the video quality experience with respect to time, and more importantly apportion the blame among multiple providers. Until $t = 5$, the load on the network is low, so the packets go through smoothly and the video player at receiver buffers the incoming frames. However, after $t = 5$, we add more traffic load entering to NSP1, so NSP1 experiences significant queueing delays increasing the jitter, and we observe glitches as "X" marks on X axis, with their respective vertical lines in Figure 4.8. Having several glitches, the buffer finally experiences an underflow and video player "freezes" at $t = 8$ and $t = 10$. At $t = 12$, we remove that additional traffic, so the load is again low till $t = 17$. At $t = 17$, the incoming traffic to NSP2 increases and this additional traffic goes to the chooser through NSP3. Note, however, that MP2 does not indicate a variation in delay, whereas the chooser experiences huge delays and jitters, causing again glitches followed by a freeze. This means, NSP2 easily handles the extra load, whereas NSP3's bottleneck link is causing the congestion and is to blame in such a scenario. After $t = 26$, this additional traffic is also removed and the user experience goes back to normal. Furthermore, Figure 4.9 shows the delays added by each NSP during the same experiment. It clearly shows NSP2 adds almost no delay under any circumstances, while NSP1 is to blame for the considerable delay between $t = 5$ and $t = 12$ and for also some variation in delay till $t = 24$. However, we can easily observe that NSP1 is not to blame for the problems encountered between $t = 17$ and $t = 26$, so NSP3 is to blame for the delay between that time interval.

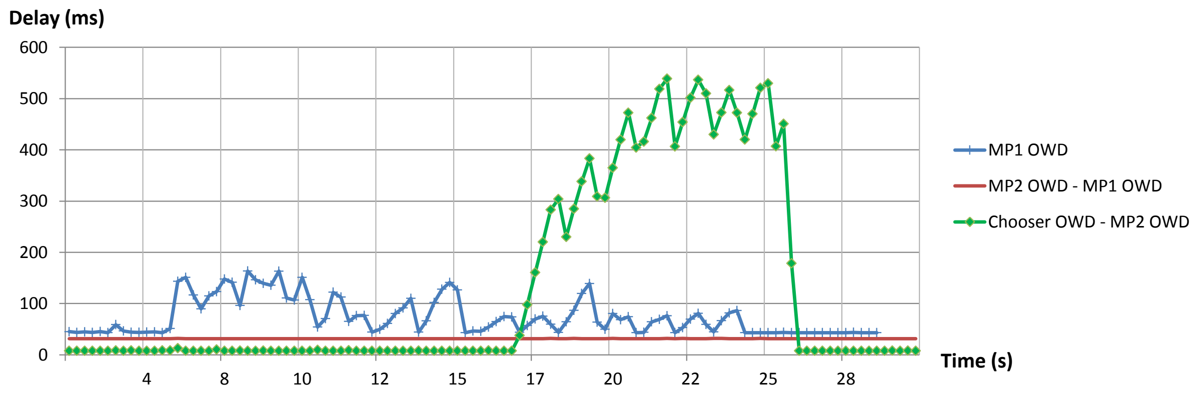


Figure 4.9: Additional delay per provider, where each owd data point represents a marker packet based measurement at each entity, and the time axis denotes the time SP sent this marker packet. Note that this figure shows the differences between providers, for example "MP2-MP1" represent the owd difference between MP2 and MP1 for each marker packet received.

While this case study clearly shows the power of such an architecture to observe the network conditions relevant to the customer's experience even with simple mechanisms, the customer might not be necessarily interested in observing and interpreting all the figures and numbers, and instead might just prefer a very short and simple explanation of each NSP's performance. MASP is the entity responsible for this task and we now provide two simple such explanations using the same MD used in Figure 4.8 and 4.9.

We first denote the "contributing delay" and "contributing jitter" in Equation 4.1 and 4.2 respectively, where d_i^k is the one-way-delay (owd) measured at node k for a marker packet with sequence number i and the contributing delay cd_i^k refers to the owd that is measured for node k, on top of the measured owd value at the previous node $k-1$, for a marker packet with sequence number i. The "contributing jitter" cj_i^k is then the difference between the last two cd values, that we use for "jitter apportionment" calculation. We also note that the first node is the sender which we assume introduces no delay difference and that is why it is always 0, i.e., $cd_i^1 = d_i^1$.

$$cd_i^k = d_i^k - d_i^{k-1} \tag{4.1}$$

$$cj_i^k = cd_i^k - cd_{i-1}^k \tag{4.2}$$

The first and simplest approach is to take the mean of the contributing jitter for the marker packets received at each provider, denoted by mcj^k at node k in Equation 4.3. Note that, because jitter is the difference between two packets, we do not count the first packet and so we have $n-1$ values. Also we take the absolute value of cj_i^k so that positive and negative jitter values do not cancel each other. We refer to this approach as "M1" and show its results with contribution percentages in Table 4.2.

$$mcj^k = \frac{\sum_{i=2}^{n}(|cj_i^k|)}{n-1} \tag{4.3}$$

Another approach is to consider that the standard deviation [164] of the jitter values, because the spikes introduce more problem to the user experience. The standard jitter contribution measured for node k, scj^k is defined in Equation 4.4. We refer to this approach as "M2" and show its results with contribution percentagess in Table 4.2.

$$scj^k = \sqrt{\frac{\sum_{i=2}^{n}(cj_i^k - mcj^k)^2}{n-1}} \tag{4.4}$$

A third approach is to simply consider the maximum jitter experienced for each provider within this experiment, i.e., $max(cj^k)$ for each k, referred as "M3" in Table 4.2.

We have only shown three basic approaches and there may obviously be several more

Table 4.2: Jitter Apportionment with three methods and their percentages

Provider	M1	M1 Percentage	M2	M2 Percentage	M3	M3 Percentage
NSP 1	16.84	44.6%	32.75	32.8%	92.4	25.3%
NSP 2	0.08	0.2%	0.27	0.3%	0.6	0.2%
NSP 3	20.82	55.2%	66.97	66.9%	272.3	74.5%

ways to provide results to the customer within the same MD, which can potentially encourage innovative new players to enter the market.

4.4 The GENI Prototype

In this section, we demonstrate the real world results of our verification service using GENI experimental facility (discussed in Appendix B.1) based upon our preliminary work [8] where we have tested a basic prototype in NS-3 simulation. We first give the experiment setup details followed by the numerical results and the evaluation of these results. In the next section, we also give the implementation details of a minimum yet complete ChoiceNet prototype with Verification service.

As we have discussed the verification service architecture in Chapter 3, Figure 4.10 shows the verification service as a set of interfaces between two roles; a "measurement provider" and an "analysis provider". Therefore we are separating the raw measurement data from the verification results, which is much more relevant to the customer. The main motivation behind designing the verification service as a set of components and their interactions is to provide flexibility. However, the design itself does not specify "what measurements to make" and "how to make measurements" since the architectural roles and interactions must be agnostic to the measurement metric and the mechanisms used. However, in practice, we would like to demonstrate that by using only one simple mechanism, the system can easily provide measurement data for a set of well-recognized metrics such as one-way delay, jitter, throughput, packet loss for each point a measurement is made. In order to achieve that, in our early work discussed in Section 4.3, we proposed a simple yet novel algorithm called the marker method that "marks" packets every so often, and each node, upon receiving the marker packet, finds the desired results by the data gathered in between markers. This can be the number of packets, bytes, the time passed or a combination of them all.

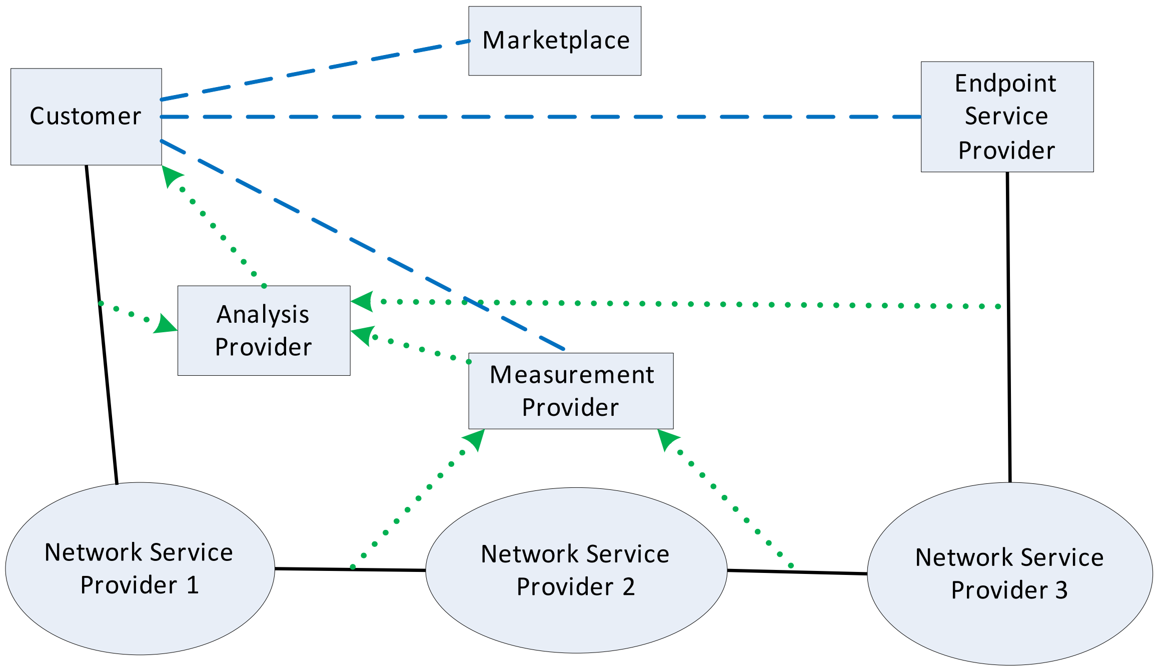

Marketplace
Customer
Endpoint Service Provider
Analysis Provider
Measurement Provider
Network Service Provider 1
Network Service Provider 2
Network Service Provider 3

Figure 4.10: Verification Architecture

4.4.1 GENI Experiment Setup

GENI ORCA control framework [132] (discussed in Appendix B.1) allows stitching several network and computer resources to provide a “slice” to the experimenter. We used Flukes GUI [57] in order to create the slice and manage its resources across two domains; BBN at Massachusetts and FIU at Florida as shown in Figure 4.11;

We have made a test case that sends a UDP stream of packets of constant size from a video server to the customer that is going over three nodes (running Ubuntu 12.04) in blue dashed arrow, each representing a service provider (NSP1, NSP2, NSP3) connected by links (LINK1, LINK2, LINK3, LINK4) of 10 Mbps and each such service provider is attached to a background node (BG1 and BG2) that sends or receives traffic in dotted green arrow, via LINK BG1 and LINK BG 2. Each measurement node is implemented as a process running at service provider nodes and each taps into the corresponding incoming interface of the service provider node, using libpcap [176] library. We envision such measurement nodes can be located at Internet Exchange Points and can be realized by optical splitters, port mirrors, tap devices or packet analyzers. We also assume the

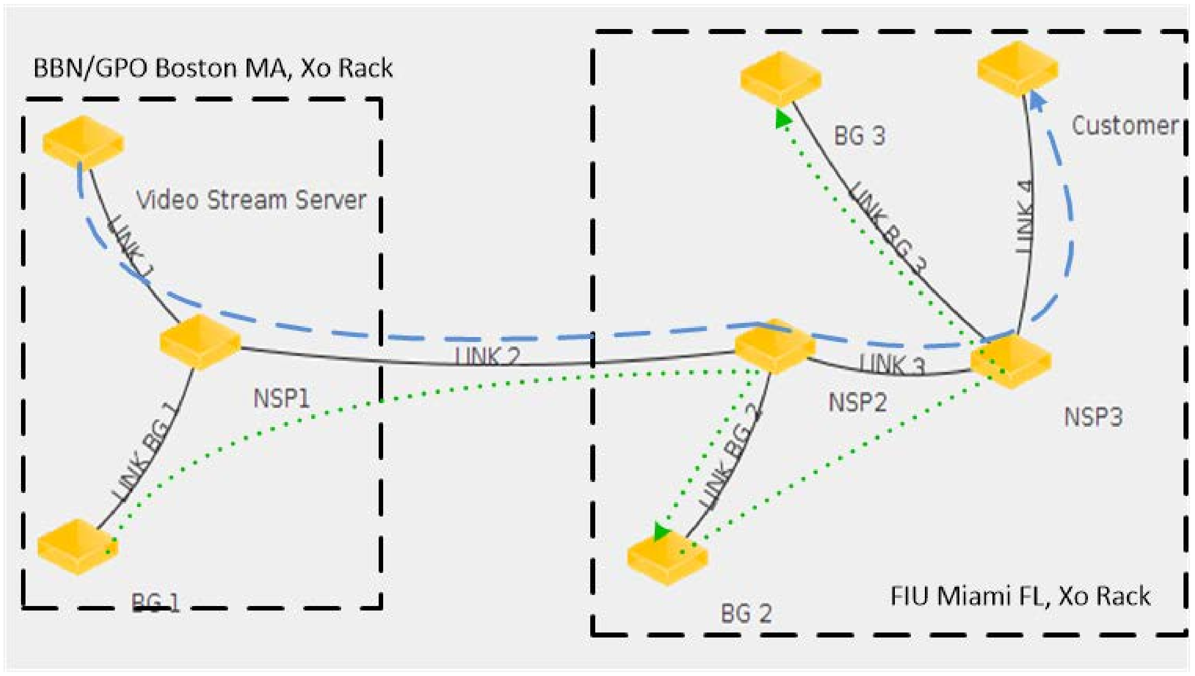


Figure 4.11: GENI Experiment Topology

path information is already provided to the measurement service (for example by a path service). We also assume all nodes are synchronized, which is hard to achieve in real life, so the results include a relatively small error.

We have used the background nodes to generate some amount of traffic on an on/off basis in order to show the effects of disruptions and allowing the measurement nodes to give us the corresponding observations for each network service provider. We have used Iperf [58] tool to generate UDP traffic. In the following section, we give the results of this experiment. Note that the experiment setup can also work for TCP traffic and with multiple flows on the same network.

4.4.2 Numerical Results

We give the results for throughput, one-way delay, jitter and packet receipt in Figure 4.12, 4.13, 4.15, 4.16 respectively. In this experiment, we have modified the video server's sending rate (the amount of packets sent in a second and the packet size) at various times. However, we only begin with the throughput observed at NSP1 since its results are exactly same as the server. In Figure 4.12, we observed the throughput behaviour is changing after NSP2 and NSP3 via high and low peaks. This is due to the fact after a relief from the congestion, some queued packets at the previous provider is sent and causes high throughput for that instant. In addition, the X axis is denoted by "marker index" as an indicator of time close to 0.5s, because in our experiment each measurement is actually performed between marker packets. Through experiments, we also note the round trip delay between NSP1 and NSP2 (between Boston and Miami) is around 40ms.

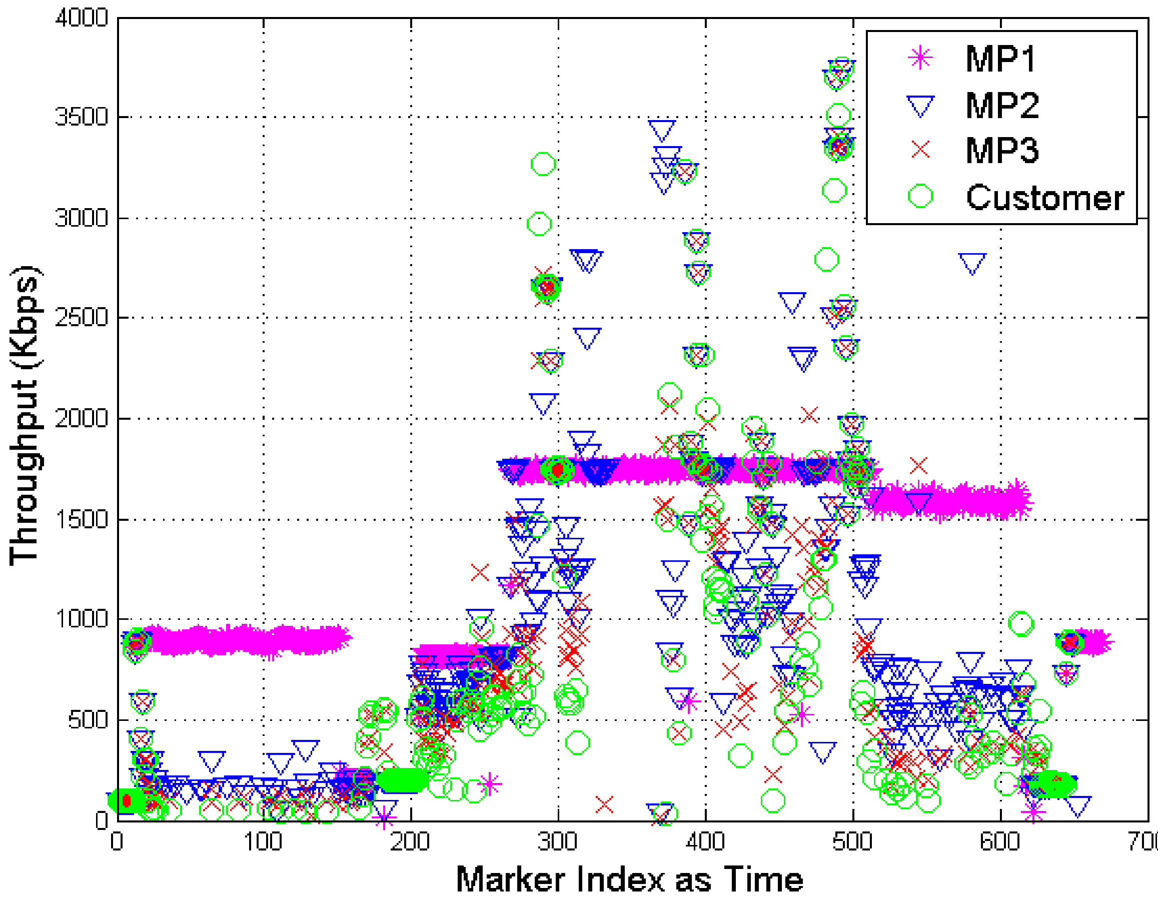


Figure 4.12: Throughput

Figure 4.13 shows the one-way delay (the time difference between a packet received a measurement node and a packet sent from the video server). The "infinity lines" in this figure and in Figure 4.14 denote the loss of marker packets to simplify interpretation. Considering the loss of consecutive marker packets is highly unlikely, we can observe a high loss rate at various times, although MP1 (purple line close to X axis) is stable. The losses are mainly due to traffic injection, however, the link connecting Boston and Miami does also have a considerable loss rate by itself (more than %10 after 2 Mbps) for especially high frequency large size UDP packets. Overall, the figure is helpful in seeing the "cumulative" effect of one-way delays, that is each measurement node adds delay and affects the observed delay at the next provider. This allows us to detect that a high delay observed at the destination does not necessarily mean the fault of the last provider and so NSP1 which adds up the considerable delay is the faulty one.

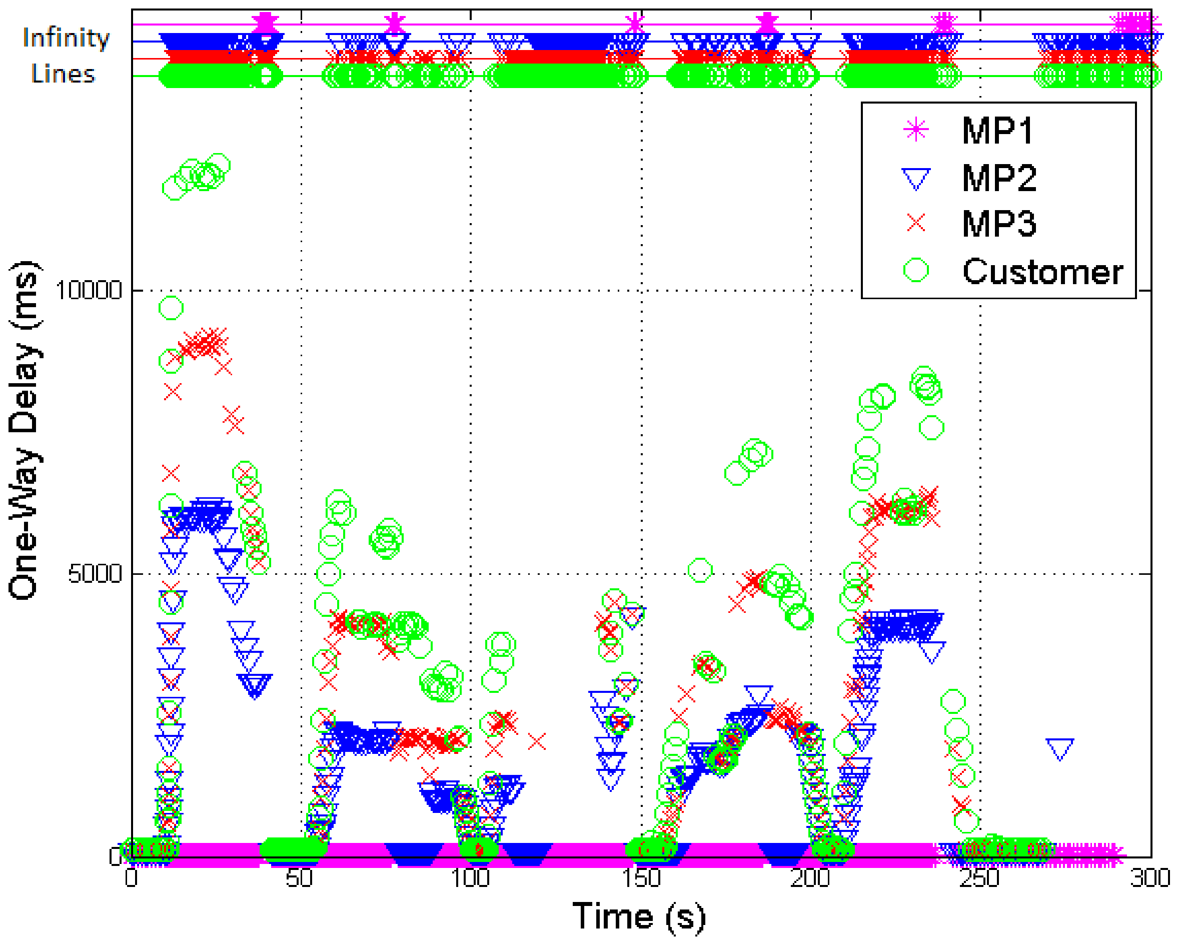


Figure 4.13: One way delay

Since Figure 4.13 show the cumulative effect of the one-way delay across providers we have also investigated the "contribution" of each provider to the one-way delay by plotting how much each provider adds up to the overall one-way delay shown in Figure 4.14. This figure gives a clear proof that NSP1 is the source of additional delay many times because the link between the "video server" and NSP1 is excellent.

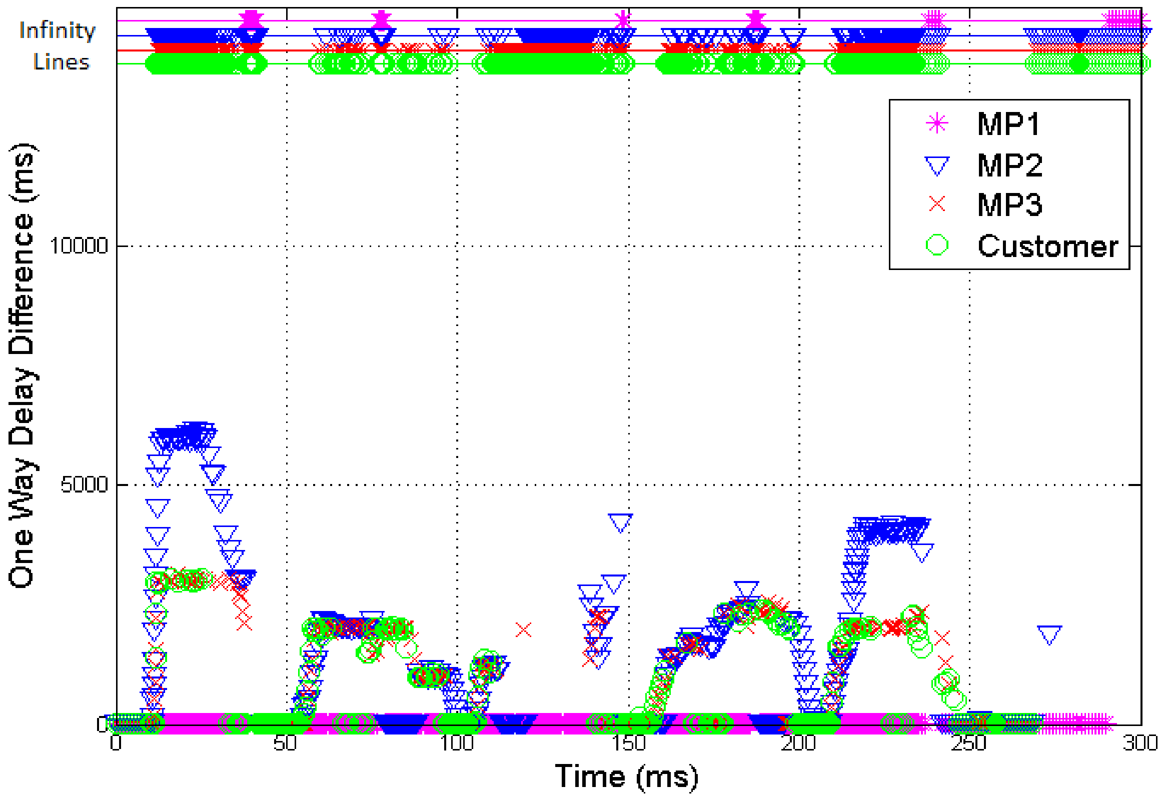


Figure 4.14: One way delay difference

As shown in Figure 4.15, the measurement service can find out the jitter (the time difference between the two last received marker packets) providing insight about the quality of service problems. In other words, a video conference or VoIP communication can be significantly affected by the jitter. Again, the spikes we have observed are mainly signs of the congestion or the relief after congestion. Also, the users are more interested in the high-level results so a number as a percentage indicating the analysis are provided in Table 4.3 for jitter and one-way-delay by taking the mean of the related metric.

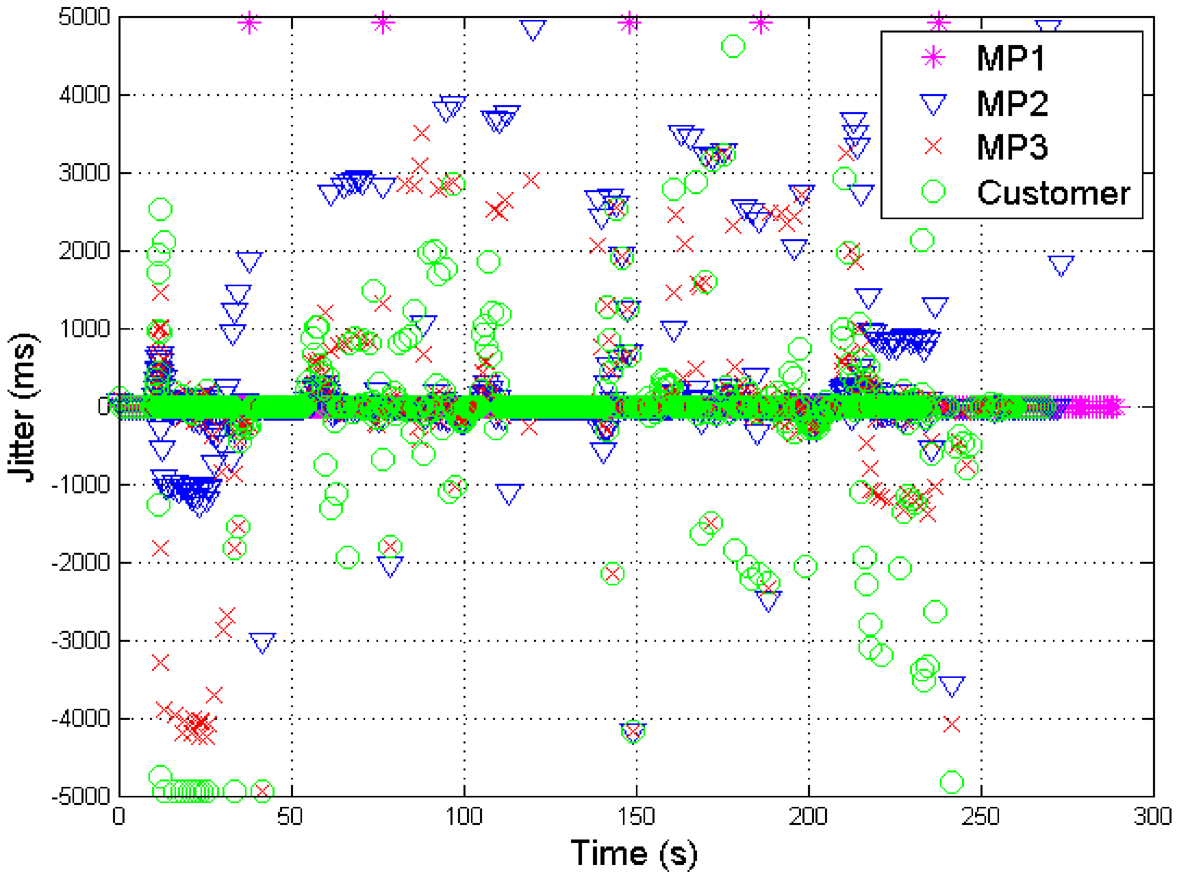


Figure 4.15: Jitter

Figure 4.16 demonstrates the number of packets received at each measurement node, between the current marker packet versus the previous marker packet. The ideal is 100 but again queueing and packet losses contribute to variations at nodes.

In order to get a real feel of the system, we also show screenshots of our implementation (described in Section 4.5) that we demonstrated the video captures in FIA Workshop [50]. Figure 4.17 shows the deployment in GENI. Also Figure 4.18 shows the initiation of the verification system by selecting the service from a text based web browser Lynx [103]. Finally, Figure 4.19 shows the measurements taken while a video is being streamed on the local network with virtual machines, since GENI virtual machines do not have graphics

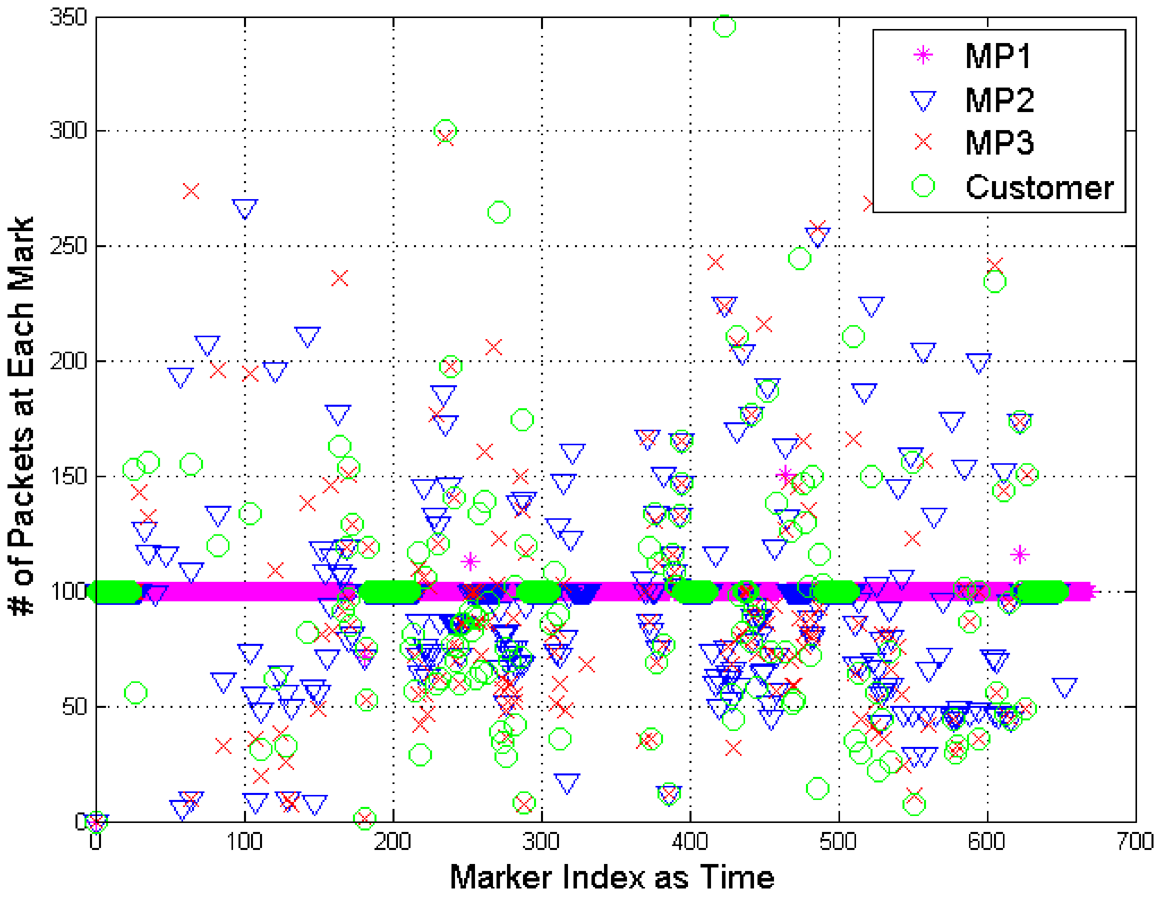


Figure 4.16: Packets received between two marker packet interval

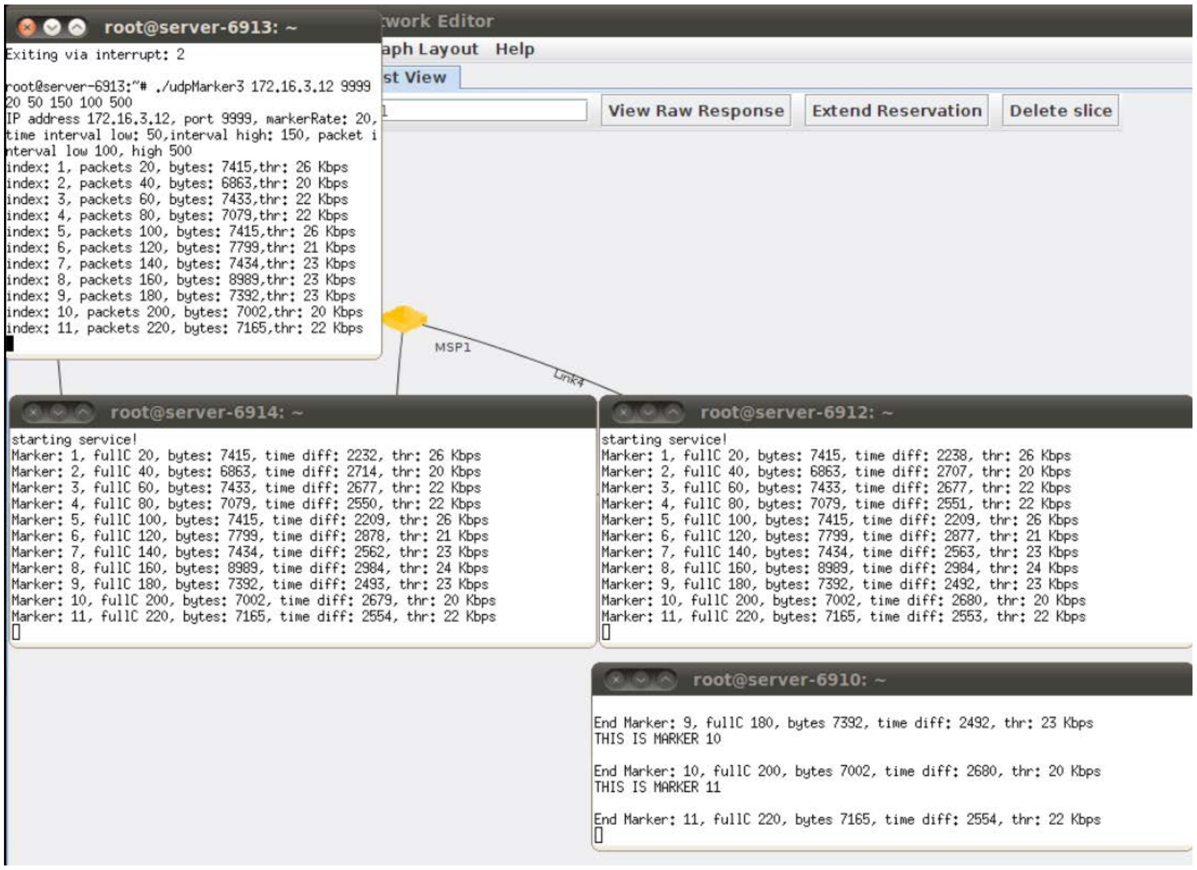
drivers installed by default.

In this section, we have shown the results of a verification service which provides feedback to the users about the performance of their services. Because such a verification service requires a complex set of interactions, we gave an overview of these roles and how they interact with each other.

The verification service needs to be tested on a real system, it also requires existence at various point of the network so we have utilized the GENI meso-scale testbed for this purpose. We have setup the experiment by allocating resources from GENI ORCA control framework and used our measurement mechanism deployed at the intermediate nodes to get the relevant user flow information. We have shown that even a simple mechanism located at the network can provide the building blocks enough to come up with several

Table 4.3: Jitter and One Way Delay Difference Apportionment by values and percentages of providers

MP	Jitter Sum Value	Jitter %	OWD Diff Sum	OWD Diff %	Marker Loss	Loss %
MP1	9936	1.7%	13.3	0.05%	40	9.4%
MP2	159100	28.2%	315002	43.3%	251	59.4%
MP3	192187	34.0%	231783	31.8%	80	18.95%
CUST	203202	36.0%	176675	24.3%	51	12%



```
Exiting via interrupt: 2

root@server-6913:~# ./udpMarker3 172.16.3.12 9999
20 50 150 100 500
IP address 172.16.3.12, port 9999, markerRate: 20,
time interval low: 50,interval high: 150, packet i
nterval low 100, high 500
index: 1, packets 20, bytes: 7415,thr: 26 Kbps
index: 2, packets 40, bytes: 6863,thr: 20 Kbps
index: 3, packets 60, bytes: 7433,thr: 22 Kbps
index: 4, packets 80, bytes: 7079,thr: 22 Kbps
index: 5, packets 100, bytes: 7415,thr: 26 Kbps
index: 6, packets 120, bytes: 7799,thr: 21 Kbps
index: 7, packets 140, bytes: 7434,thr: 23 Kbps
index: 8, packets 160, bytes: 8989,thr: 23 Kbps
index: 9, packets 180, bytes: 7392,thr: 23 Kbps
index: 10, packets 200, bytes: 7002,thr: 20 Kbps
index: 11, packets 220, bytes: 7165,thr: 22 Kbps
```

```
starting service!
Marker: 1, fullC 20, bytes: 7415, time diff: 2232, thr: 26 Kbps
Marker: 2, fullC 40, bytes: 6863, time diff: 2714, thr: 20 Kbps
Marker: 3, fullC 60, bytes: 7433, time diff: 2677, thr: 22 Kbps
Marker: 4, fullC 80, bytes: 7079, time diff: 2550, thr: 22 Kbps
Marker: 5, fullC 100, bytes: 7415, time diff: 2209, thr: 26 Kbps
Marker: 6, fullC 120, bytes: 7799, time diff: 2878, thr: 21 Kbps
Marker: 7, fullC 140, bytes: 7434, time diff: 2562, thr: 23 Kbps
Marker: 8, fullC 160, bytes: 8989, time diff: 2984, thr: 24 Kbps
Marker: 9, fullC 180, bytes: 7392, time diff: 2493, thr: 23 Kbps
Marker: 10, fullC 200, bytes: 7002, time diff: 2679, thr: 20 Kbps
Marker: 11, fullC 220, bytes: 7165, time diff: 2554, thr: 22 Kbps
```

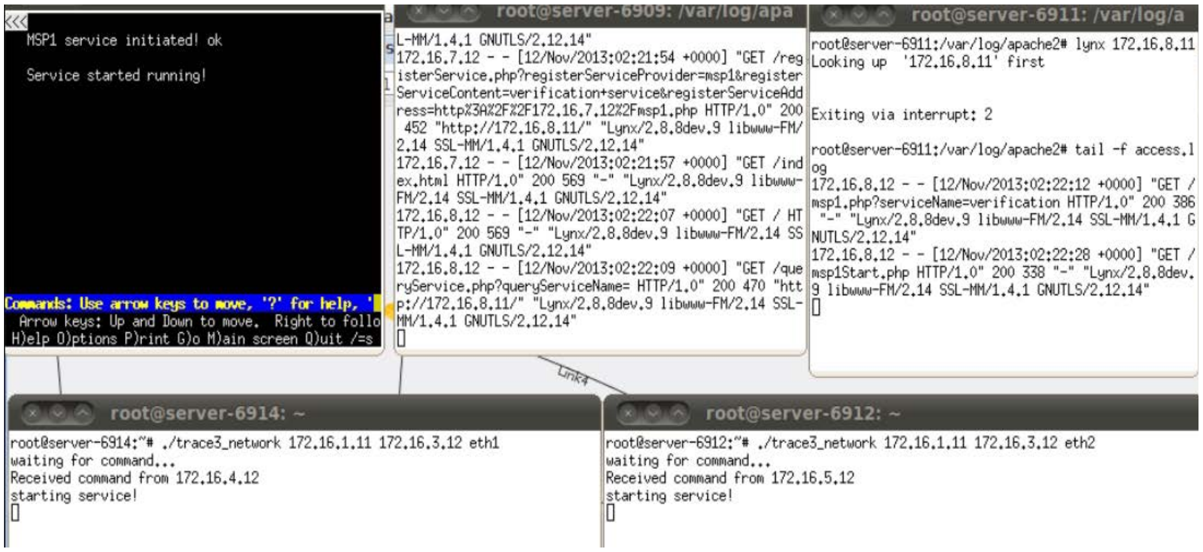
```
starting service!
Marker: 1, fullC 20, bytes: 7415, time diff: 2238, thr: 26 Kbps
Marker: 2, fullC 40, bytes: 6863, time diff: 2707, thr: 20 Kbps
Marker: 3, fullC 60, bytes: 7433, time diff: 2677, thr: 22 Kbps
Marker: 4, fullC 80, bytes: 7079, time diff: 2551, thr: 22 Kbps
Marker: 5, fullC 100, bytes: 7415, time diff: 2209, thr: 26 Kbps
Marker: 6, fullC 120, bytes: 7799, time diff: 2877, thr: 21 Kbps
Marker: 7, fullC 140, bytes: 7434, time diff: 2563, thr: 23 Kbps
Marker: 8, fullC 160, bytes: 8989, time diff: 2984, thr: 24 Kbps
Marker: 9, fullC 180, bytes: 7392, time diff: 2492, thr: 23 Kbps
Marker: 10, fullC 200, bytes: 7002, time diff: 2680, thr: 20 Kbps
Marker: 11, fullC 220, bytes: 7165, time diff: 2553, thr: 22 Kbps
```

```
End Marker: 9, fullC 180, bytes 7392, time diff: 2492, thr: 23 Kbps
THIS IS MARKER 10

End Marker: 10, fullC 200, bytes 7002, time diff: 2680, thr: 20 Kbps
THIS IS MARKER 11

End Marker: 11, fullC 220, bytes 7165, time diff: 2554, thr: 22 Kbps
```

Figure 4.17: Screenshot for measurement points from GENI deployment

metrics at each measurement point, which can easily detect each provider's contribution to performance problems.

We envision such building blocks of measurement data can be better utilized via


```
<<<
  MSP1 service initiated! ok

  Service started running!

Commands: Use arrow keys to move, '?' for help, '
  Arrow keys: Up and Down to move.  Right to follo
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=s
```

root@server-6909: /var/log/apa

```
L-MM/1.4.1 GNUTLS/2.12.14"
172.16.7.12 - - [12/Nov/2013:02:21:54 +0000] "GET /reg
isterService.php?registerServiceProvider=msp1&register
ServiceContent=verification+service&registerServiceAdd
ress=http%3A%2F%2F172.16.7.12%2Fmsp1.php HTTP/1.0" 200
 452 "http://172.16.8.11/" "Lynx/2.8.8dev.9 libwww-FM/
2.14 SSL-MM/1.4.1 GNUTLS/2.12.14"
172.16.7.12 - - [12/Nov/2013:02:21:57 +0000] "GET /ind
ex.html HTTP/1.0" 200 569 "-" "Lynx/2.8.8dev.9 libwww-
FM/2.14 SSL-MM/1.4.1 GNUTLS/2.12.14"
172.16.8.12 - - [12/Nov/2013:02:22:07 +0000] "GET / HT
TP/1.0" 200 569 "-" "Lynx/2.8.8dev.9 libwww-FM/2.14 SS
L-MM/1.4.1 GNUTLS/2.12.14"
172.16.8.12 - - [12/Nov/2013:02:22:09 +0000] "GET /que
ryService.php?queryServiceName= HTTP/1.0" 200 470 "htt
p://172.16.8.11/" "Lynx/2.8.8dev.9 libwww-FM/2.14 SSL-
MM/1.4.1 GNUTLS/2.12.14"
```

root@server-6911: /var/log/a

```
root@server-6911:/var/log/apache2# lynx 172.16.8.11
Looking up  '172.16.8.11' first


Exiting via interrupt: 2

root@server-6911:/var/log/apache2# tail -f access.l
og
172.16.8.12 - - [12/Nov/2013:02:22:12 +0000] "GET /
msp1.php?serviceName=verification HTTP/1.0" 200 386
 "-" "Lynx/2.8.8dev.9 libwww-FM/2.14 SSL-MM/1.4.1 G
NUTLS/2.12.14"
172.16.8.12 - - [12/Nov/2013:02:22:28 +0000] "GET /
msp1Start.php HTTP/1.0" 200 338 "-" "Lynx/2.8.8dev.
9 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/2.12.14"
```

root@server-6914: ~

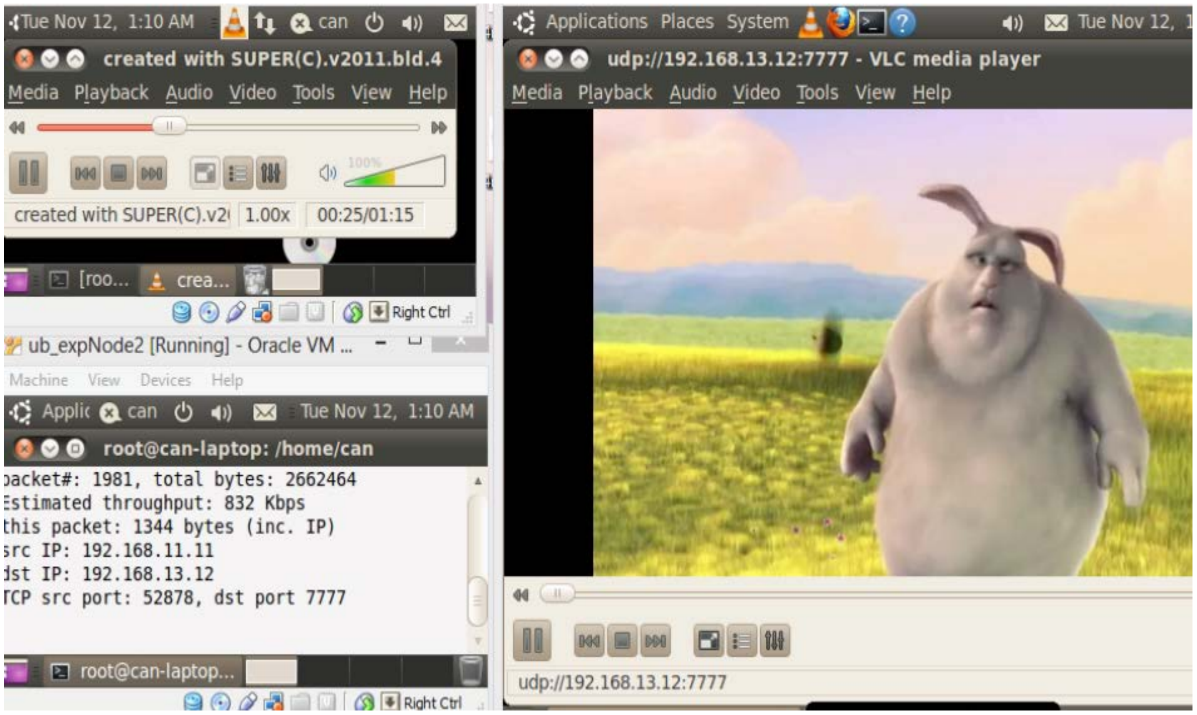
```
root@server-6914:~# ./trace3_network 172.16.1.11 172.16.3.12 eth1
waiting for command...
Received command from 172.16.4.12
starting service!
```

root@server-6912: ~

```
root@server-6912:~# ./trace3_network 172.16.1.11 172.16.3.12 eth2
waiting for command...
Received command from 172.16.5.12
starting service!
```

Figure 4.18: Screenshot of service initiation from GENI deployment

complex analytical analysis. Also the availability of GENI to provide "on-demand" slices consisting of geographically distributed nodes allow us to perform our tests for various cases at a real system, improves repeatability of the experiment on a real distributed network and provide more value to the research.


Tue Nov 12, 1:10 AM
can
created with SUPER(C).v2011.bld.4
Media Playback Audio Video Tools View Help
created with SUPER(C).v2
1.00x
00:25/01:15
[roo...
crea...
Right Ctrl
ub_expNode2 [Running] - Oracle VM ...
Machine View Devices Help
Applic
can
Tue Nov 12, 1:10 AM
root@can-laptop: /home/can
root@can-laptop...
Right Ctrl
Applications Places System
Tue Nov 12,
udp://192.168.13.12:7777 - VLC media player
Media Playback Audio Video Tools View Help
udp://192.168.13.12:7777

```
packet#: 1981, total bytes: 2662464
Estimated throughput: 832 Kbps
this packet: 1344 bytes (inc. IP)
src IP: 192.168.11.11
dst IP: 192.168.13.12
TCP src port: 52878, dst port 7777
```

Figure 4.19: Screenshot of video stream measurement at local network

4.5 The GENI Prototype Design

As of the writing of this thesis, a comprehensive ChoiceNet project prototype is still an ongoing effort, however, we have made contribution to the design and implementation of a minimum yet complete ChoiceNet prototype with verification service and in this section, we provide details of it by first giving a high level view of the each individual task to be completed, followed by a discussion on the implementation details of each role.

4.5.1 High Level Task Plan

1. Specification of the communication protocol and exchanged information:

    - The transport (or message exchange) protocol for use plane and economy plane
    - The format, content and semantics of the information exchanged.
    - An example service definition

- The additional information (payment, security, id) to be kept at NULL (empty) and yet supported by the protocol.

* The transport protocol is based on SOAP messages for the economy plane and a (flexible) specification at service definition for use plane.

2. ChoiceNet Basic API:
   The definition and implementation for a common set of APIs in ChoiceNet that will make a call to a “well-known” marketplace to “query” / “insert” / “update” service in the marketplace, as well as to provider such as “request” service from provider, “reply” a service, get status of a service etc. These APIs will use the communication protocol defined above to exchange information. This does not exclude the payment or identity or security related issues but they can be parameters or APIs as NULL.

3. Marketplace Related functionality:

   - A simple server as marketplace that gets the communication requests through standard APIs and modifies the database accordingly. This data includes the service definitions, provider information and the providers service information.
   - Querying the marketplace for a service, means searching the services offered by providers, which can be a simple text search to an advanced, intelligent search. We believe a basic search is fine for the minimal prototype.

4. Provisioning part of Path Service as a service example:
   Since a provider must “use” a resource to provide a “service”, we would like to show how a provider can dynamically provision of path resources (links and network device flow priority) to make sure the service guarantees are met. The interactions for this will be between a provider and its network devices, thus we do not need a ChoiceNet standard, but it is good to provide a well-known standard than creating our own.

   * Software Defined Networking is a good candidate to enable path provisioning, but we should also think about customer-related requirements such as tokens inside packets.

5. High Level Role Implementation:

- This section utilizes the ChoiceNet API to achieve the customer or provider functionality. For a customer, an application process is needed to use ChoiceNet API to query the services and then request a service from a provider. For a provider, a provider process waits for customer requests and understands the request and provisions it according. Again we can put NULL for identity, security and payment.
- A "monitor" tool is necessary for debugging, that lets us manually execute a ChoiceNet API with the parameters or see what the status of the customer or provider is.

6. Complex Service Definition and Planning Service:

    - Creating service definitions for popular use-cases, and showing how these services can be bundled to come up with a plan (or recipe).
    - This also includes the service definition of the planning service itself.
    - There also should be a corresponding planner related program at customer that "understands" this recipe.

7. Verification Service Example:
    The verification service itself consists of two components; a measurement provider and an analysis provider. The measurement provider needs to get measurement data from end nodes and intermediate nodes. The tasks in this context are;

    - Design/implement a measurement mechanism (such as the marker method) to extract measurement from all nodes by various ways (either a software tap such as libpcap at the same node or via port mirroring). This process should also be able to get parameters (locally or remotely at the customer/provider) and apply it dynamically.
    - Design a "measurement data format" to wrap the raw measurement, and send it to measurement analyzer. This should be a standard and composable format (or measurement specification) and part of the service definition
    - The analyzer will then compare the requirements with the measured data and will give result(s) to the customer. The result format must be designed. This should also be part of the service definition.

- Also a basic tool that shows us the end results in real-time or off-line. A basic graphical interface to show figure would be nice.

8. Real Life Applications demanding QoS:
   The popular tools at end points to demonstrate the practical benefits of ChoiceNet, especially for real time user experience. For example;

   - Video streaming application to get the user experience in a quantified way (pauses, pixel losses etc)
   - Video Conferencing implementation; or an implementation that imitates video conferencing and gets quantifiable user experience results.

   *In order to get the statistic data, these tools may need to be modified, for example VLC.

9. Computing and Networking Resources to Realize ChoiceNet:

   - GENI-ORCA is a good candidate to realize ChoiceNet via virtual machines connected together through tunnels.
   - At a later stage, real Openflow switches can be used that are controlled by ChoiceNet for path service, and NetFPGA like boxes to implement measurement capability at high loads.

4.5.2 Implementation Details for Roles

We now specify the implementation details for each role, which includes the subcomponents and interactions.

The Chooser

Figure 4.20 shows a general view of the implementation at customer side. ChoiceNet functions at the customer are called either directly by user application or by a ChoiceNet control application. This in turn (similar to DNS) triggers a set of queries to marketplace and returns results. Then the application "purchase" the service and wakes up the verification agent, which in turn taps into incoming/outgoing interfaces and sends measurements to an measurement analysis server (MASP). In practice, it can be implemented as shown in Figure 4.21.

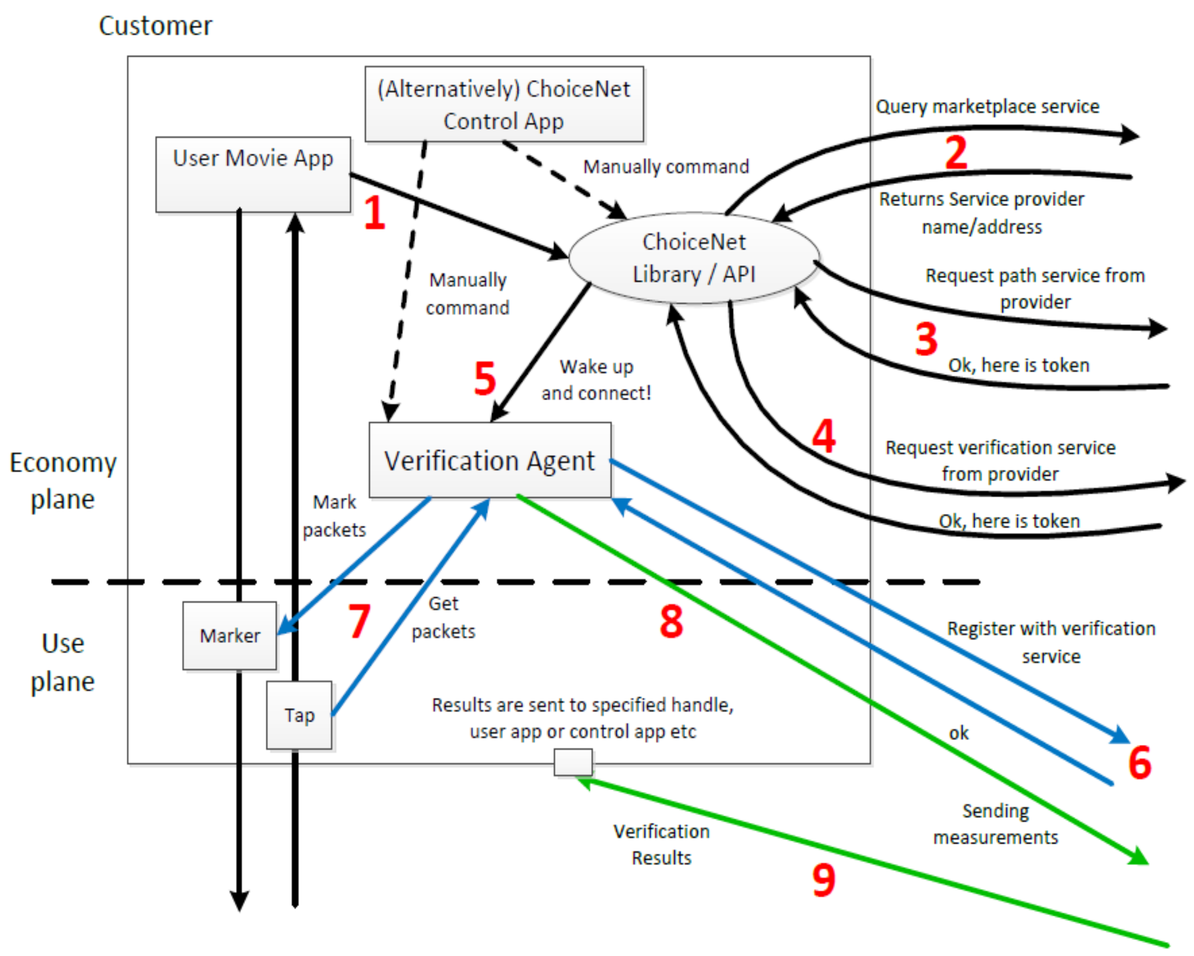

Customer
(Alternatively) ChoiceNet Control App
User Movie App
Manually command
1
ChoiceNet Library / API
Manually command
Query marketplace service
2
Returns Service provider name/address
Request path service from provider
3
Ok, here is token
5
Wake up and connect!
4
Request verification service from provider
Ok, here is token
Economy plane
Verification Agent
Mark packets
Use plane
Marker
7
Get packets
8
Register with verification service
Tap
Results are sent to specified handle, user app or control app etc
ok
6
Sending measurements
Verification Results
9

Figure 4.20: Customer Role General View

The Marketplace

As shown in Figure 4.22, in our design, the marketplace is (and is supposed to be) very simple, it is just a database lookup and update, anything else can be represented as a separate service.

Path Provider

As shown in Figure 4.23, a Path Provider entity keeps user information, whereas a master process is responsible for communication with each network device and keeps track of network resource allocation.

In practice, we can enable this functionality with a single “path provider process box”

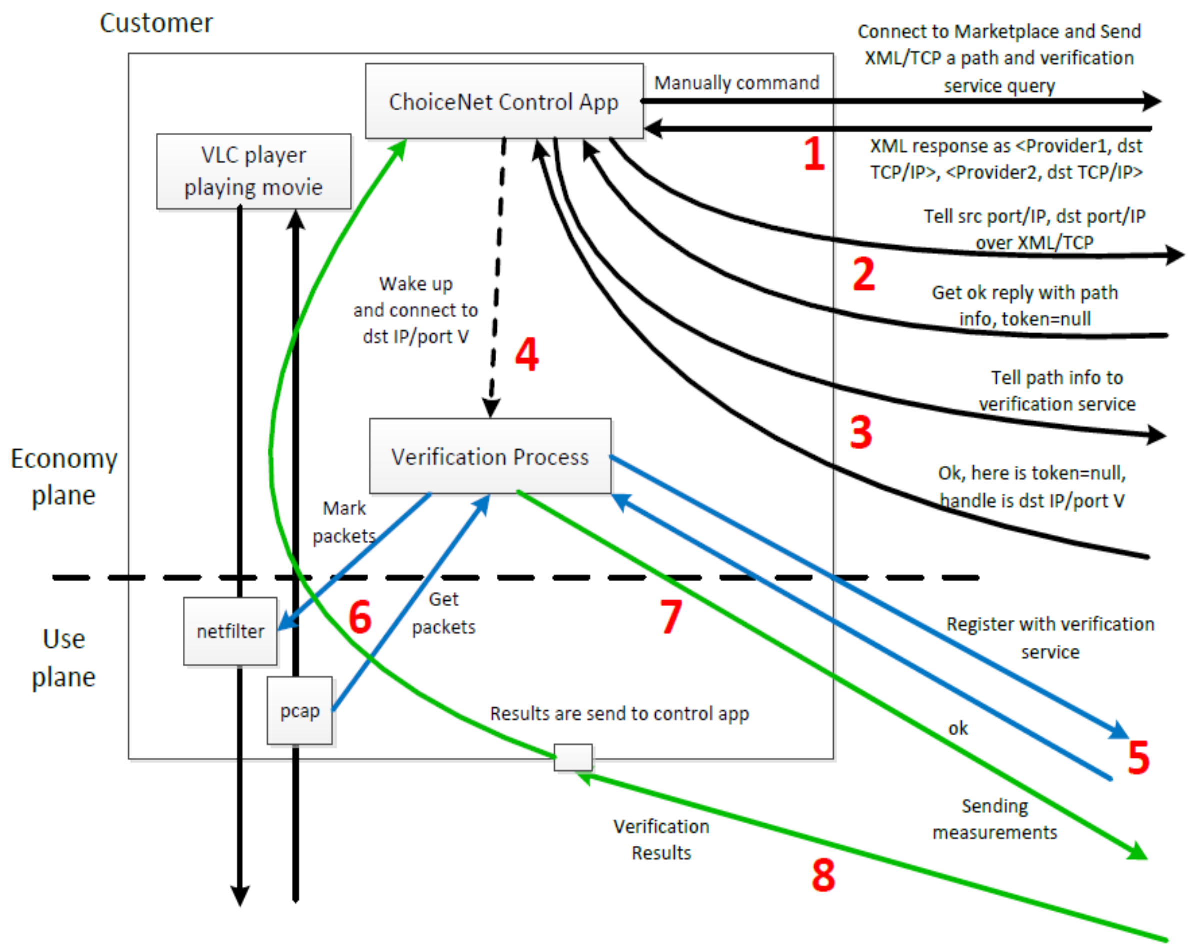

Customer
Connect to Marketplace and Send XML/TCP a path and verification service query
ChoiceNet Control App
Manually command
VLC player playing movie
1
XML response as <Provider1, dst TCP/IP>, <Provider2, dst TCP/IP>
Tell src port/IP, dst port/IP over XML/TCP
Wake up and connect to dst IP/port V
2
Get ok reply with path info, token=null
4
Tell path info to verification service
3
Economy plane
Verification Process
Ok, here is token=null, handle is dst IP/port V
Mark packets
Get packets
6
7
Use plane
netfilter
Register with verification service
pcap
Results are send to control app
ok
5
Sending measurements
Verification Results
8

Figure 4.21: The Customer Role in practice

with a single database and keep communicating with Linux boxes (for instance in GENI) with a TCP connection, where Linux boxes behave as a slave (taking commands such as "enable routing for this IP address" via IP Tables) as shown in Figure 4.24.

Measurement Service Provider

Similar to the path provider, MSP keeps track of Linux boxes that does the measurement via libpcap to filter relevant packets, but also at each box, packets are processed through some algorithm (for example, the marker method, discussed in Section 4.3). The results are returned to a "handler", which is typically a measurement analysis provider as shown in Figure 4.25.

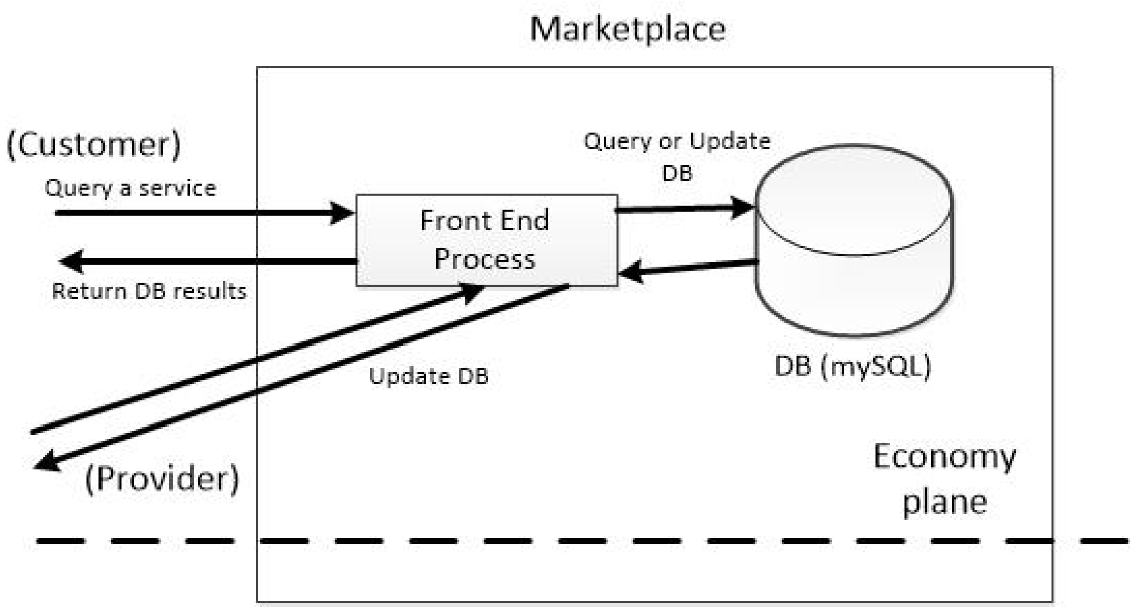


Figure 4.22: The Marketplace Role in practice

Measurement Analysis Service Provider

The Measurement Analysis Service Provider (MASP) gets the measurement data, typically stores it temporarily and then processes it to return the verification results. All measurement data is sent to this component to get the verification result as shown in Figure 4.26.

Measurement Composition Service Provider

The Measurement Composition Service Provider receives a request from a customer and based on that, it queries the marketplace and possibly the actual providers, to come up with a recipe.

End Service Provider

The End Service Provider (SP) is similar to the customer, but its verification functionality is triggered by the customer, and the verification results are not returned to it, as shown in Figure 4.28.

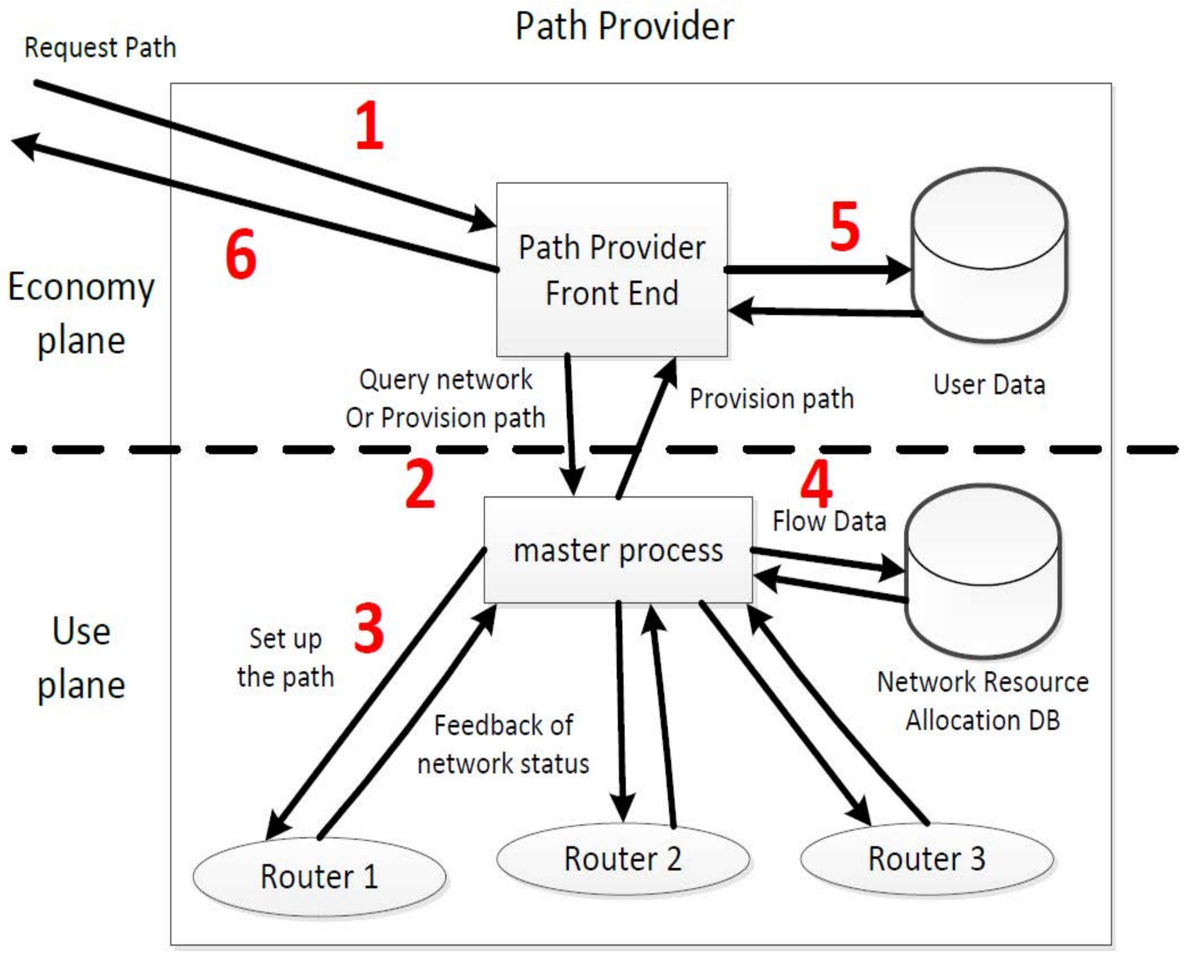


Figure 4.23: Path Provider Role in High Level View

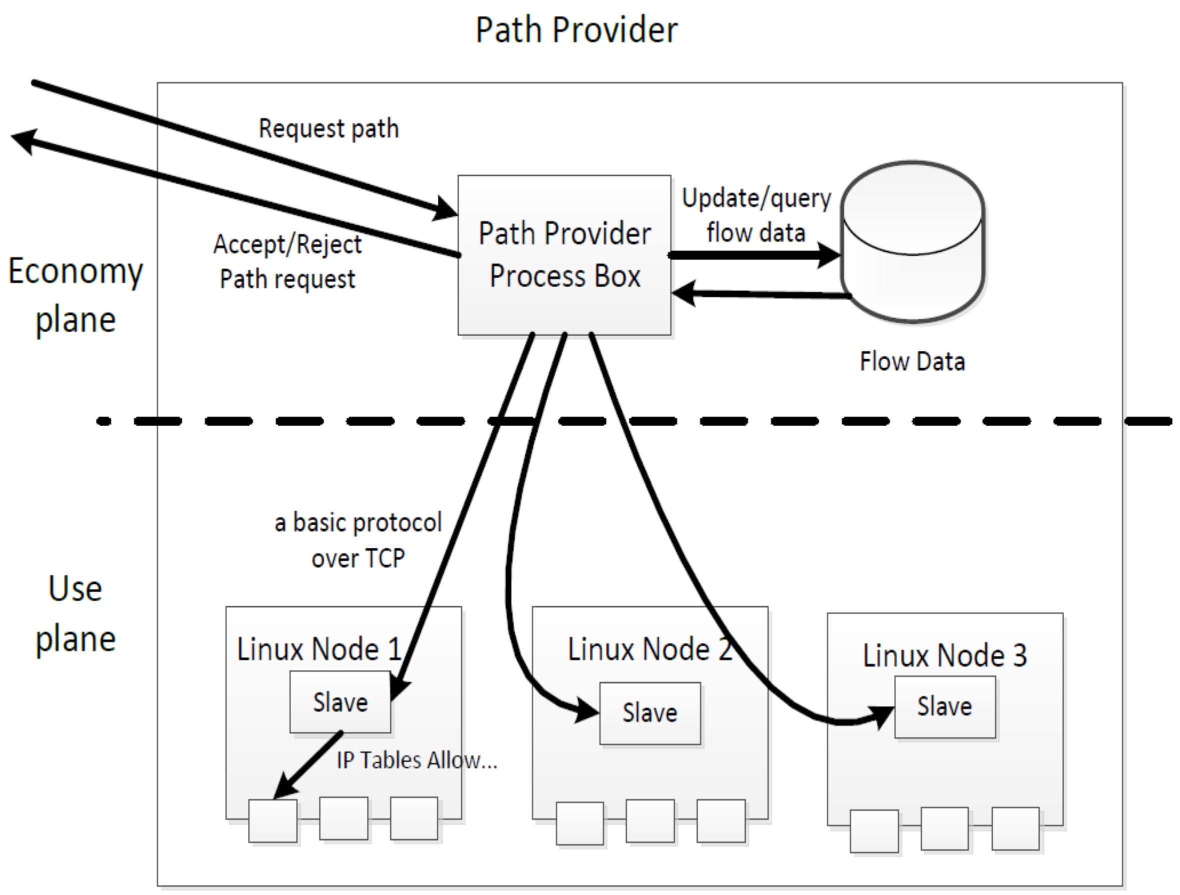


Figure 4.24: Path Provider Role in Practice

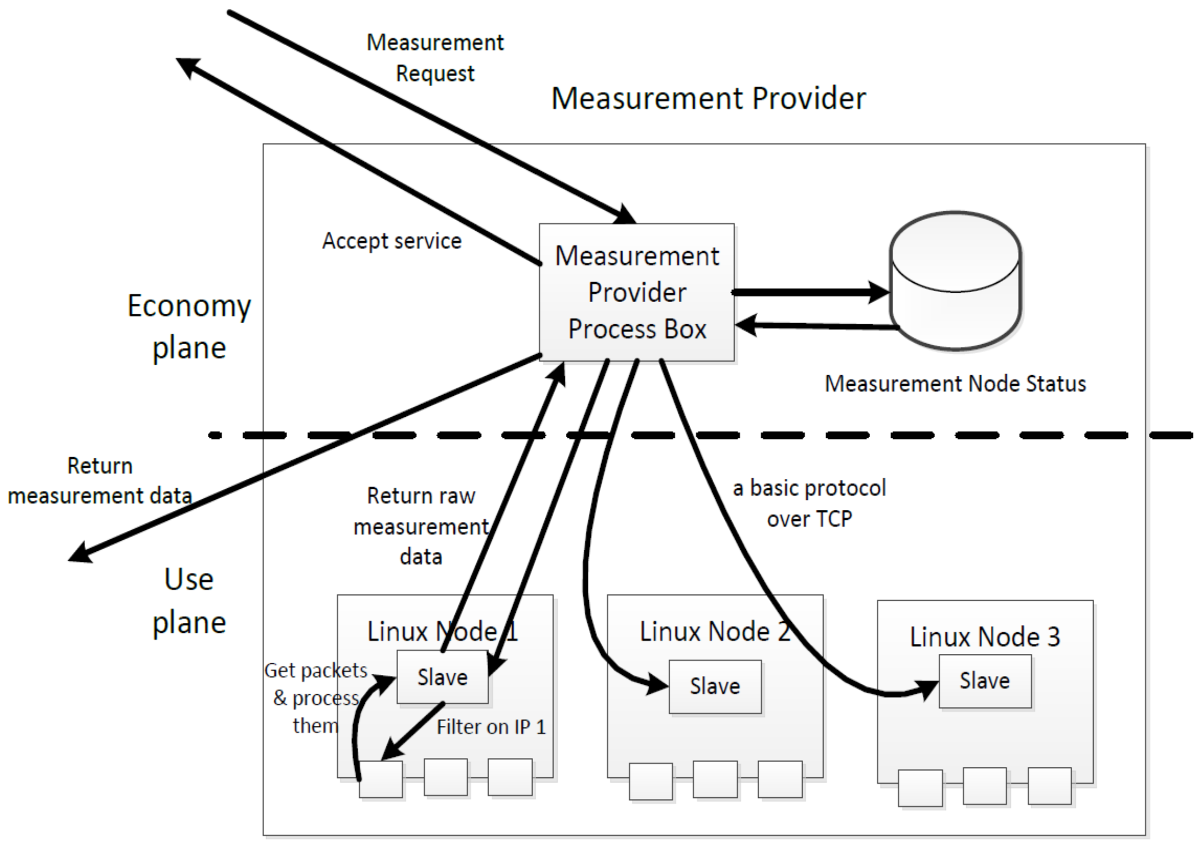


Figure 4.25: Measurement Service Provider Role in Practice

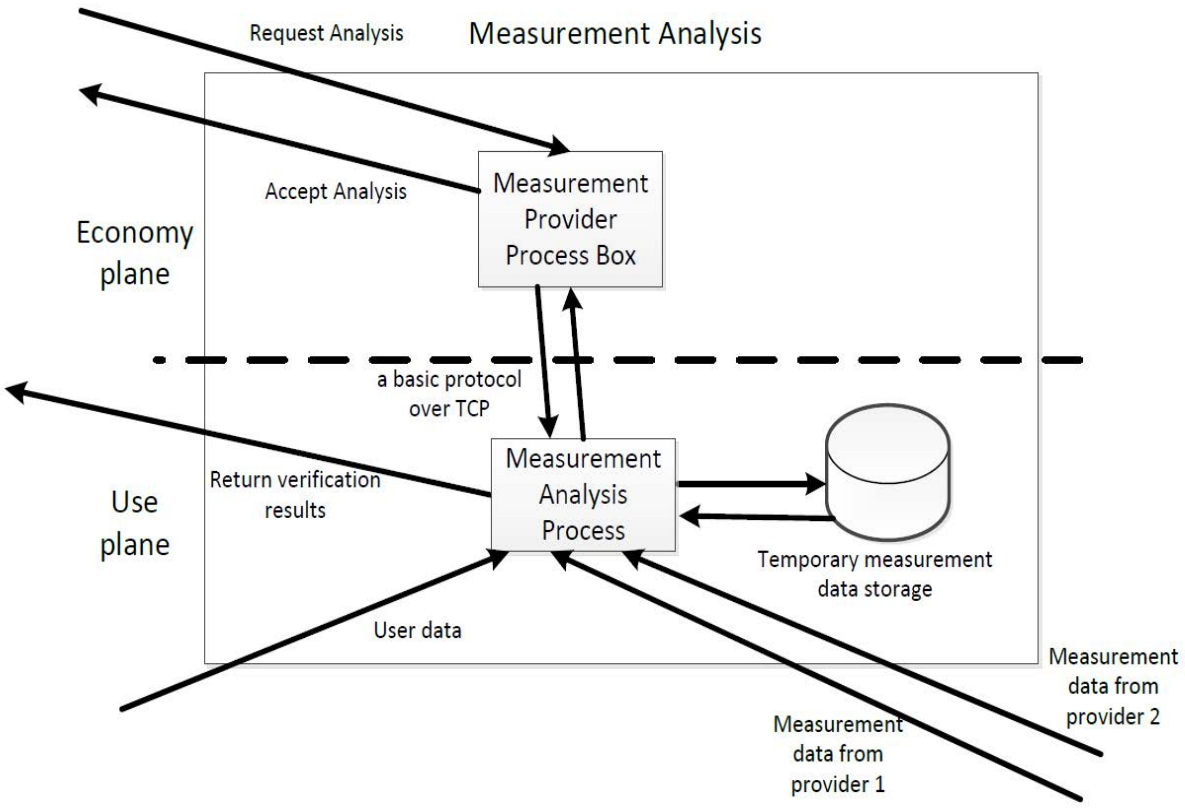

Request Analysis
Measurement Analysis
Accept Analysis
Measurement Provider Process Box
Economy plane
a basic protocol over TCP
Use plane
Return verification results
Measurement Analysis Process
Temporary measurement data storage
User data
Measurement data from provider 2
Measurement data from provider 1

Figure 4.26: Measurement Analysis Service Provider Role in Practice

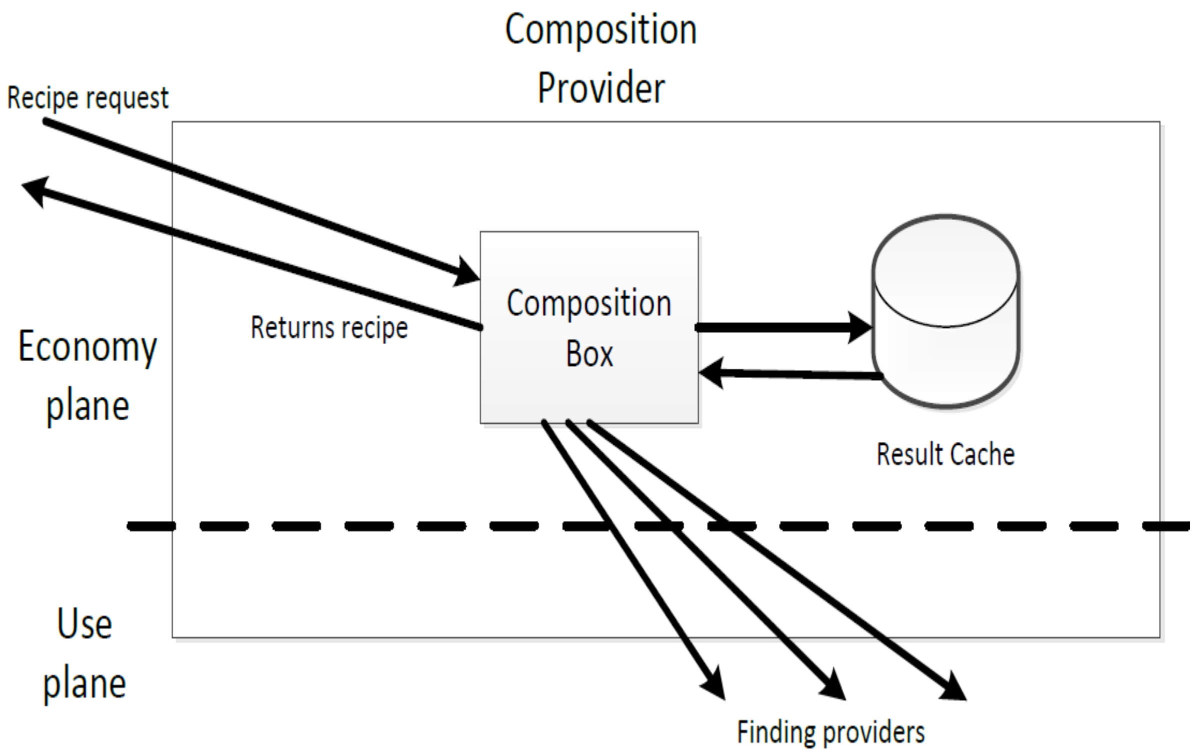


Figure 4.27: Measurement Composition Service Provider Role in Practice

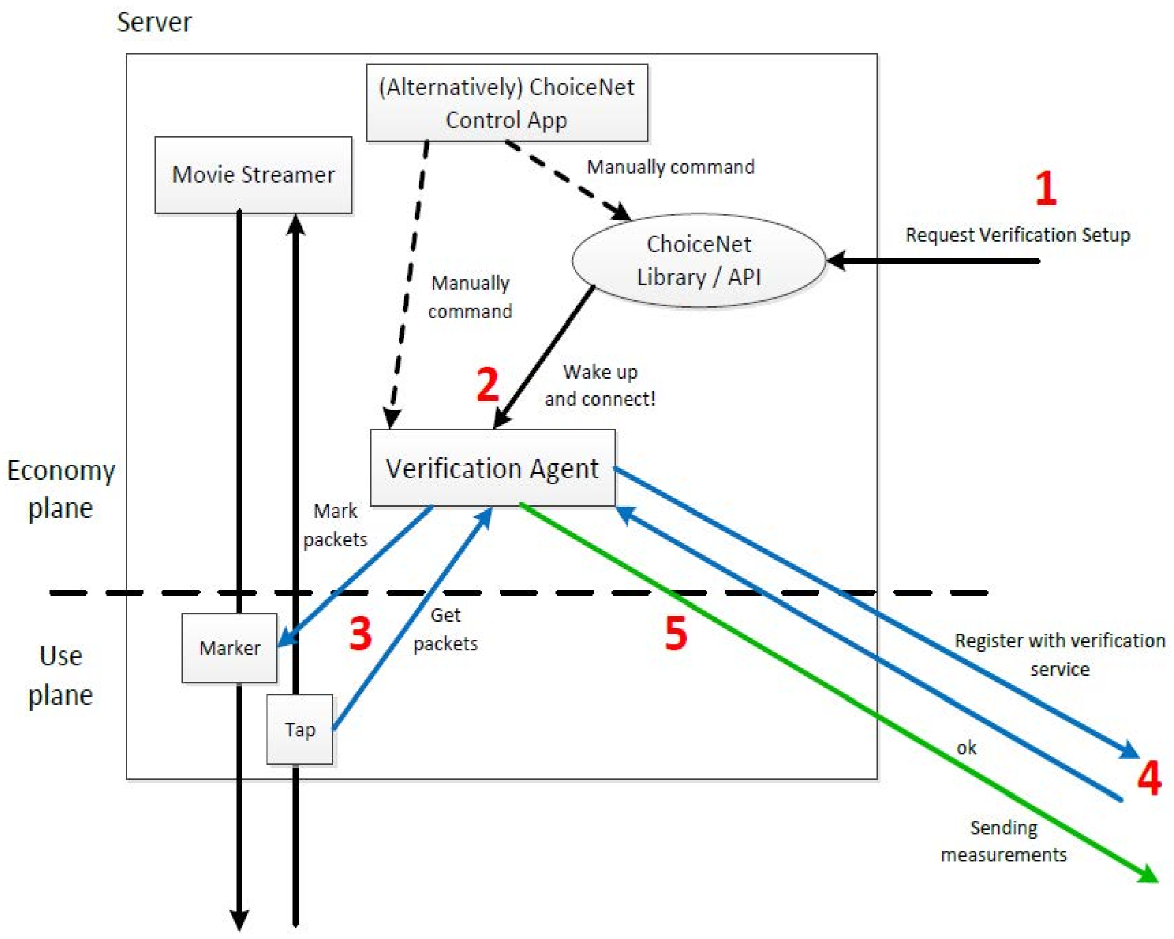


Figure 4.28: End Service Provider Role in Practice

Chapter 5

Study of Informed Choice Effect

Having explained the architectural design in detail in Chapter 3 as well as the performance of this design in Chapter 4, in this chapter, we investigate the question of whether a system that offers multiple choices to user can actually result in a "better" system, where "better" means more profit at the provider and "better" quality of service at the user. Our study is based on a practical and useful domain, which is the consideration of energy-efficiency on establishing on-demand optical paths which at the same time tries to maximize the user experience for the users who prefer high quality of service. This case study is especially practical and useful because the problem of providing an agile, energy-aware, flexible optical network architecture is one of the challenges in optical networking in the coming decade and a key element in this challenge is the balancing of the benefits to customer and provider, and creating an agile system capable of reflecting both provider and customer interests on an ongoing basis as network conditions change. Our basic premise is that a *marketplace* (discussed in Chapter 3) that allows providers to advertise services on a dynamic basis, with accompanying prices that reflect the relative resource scarcity in the network at the time, provides a rendezvous of customer and provider interests, allowing collaborative optimization of network resource usage. With these pre-announced information, customers are able to make informed choices, thus participating in optimizing the use of available network resources for win-win benefits for providers and customers. We show how such marketplace choices naturally arise for energy-aware optical networks, and investigate the ensuing customer-provider interactions. Our results demonstrate the potential for win-win in this approach.

We first give the background about the energy efficiency in optical networks along with the formulation to represent the parameters used in energy efficiency, following an earlier work of our research group [71]. We then introduce our choice-based approach for "minimum energy cost", "minimum delay" and other mixed choices. We finally describe the simulation and give an evaluation of the results.

5.1 Energy Efficiency in Optical Networks

Optical transport networks have formed the lowest level of the core of the planetary communication networks, whether voice or data, over the last decades, and can be expected to do so indefinitely into the future. Access networks and local networks continue to use electronic equipment suitable for complex computation in the path of data flows, and connect to the core optical networks using opto-electronic devices. The explosive growth of mobile networking has focused attention on wireless networks, while at the same time increasing and underscoring the dependence of planetary communications on high-bandwidth optical cores.

The optical transport must support an increasingly larger range of bandwidth needs, over more diverse timescales. As dynamic provisioning algorithms operate on heuristic principles, over time the succession of traffic demand arrival and departure can create suboptimalities in network resource utilization; thus the network must re-optimize itself from time to time. As timescales decrease, this can happen over short intervals such as hours, and the re-optimization must become an integral part of network operation, rather than a separate network management function. Finally, to allow a diverse and stable ecosystem of network operators and service providers, different niches for collaborating and competing businesses must exist, such that different business entities with different value proposition, business models, risk tolerance etc. can leverage each others' offerings to present a rich set of offerings to the customer.

A key potential contribution of optical networks lies in the fact that optical switching and transport is far more energy-friendly, bit-for-bit, than electronic forwarding options. This has given rise to "green grooming" approaches in literature as we discuss below. However, a significant practical barrier to the adoption of any such approach is the perceived loss in performance, which is considered unacceptable to providers and customers alike. In theory, every customer professes that the network should be energy-efficient and eco-friendly: in practice, customers are unwilling to accept degraded service for themselves after having paid for it. In part, the problem arises from the classical model of transport network service as being characterized by its bandwidth, and possibly other QoS elements such as delay and jitter bounds, and carrying a price tag. Factors with variable effects on the cost to the provider, such as distance traversed by the demand (with ensuing costs in amplifiers or other optical equipment to increase reach), the channel to be used (again, channels in different bands may require different equipment at line termination and at intermediate points), and, most importantly in the current context, the energy and cooling requirement of electronic equipment in the path, are not visible to the customer in the price.

We advance a *choice-based cooperative network optimization approach* that can overcome this problem. Briefly, we draw on the future Internet project *ChoiceNet* (discussed in Chapter 3 and papers in [189, 151, 8]) and its concept of a *marketplace* in which a network provider can advertise service offerings with stated SLA characteristics. Traffic grooming techniques can be used proactively by the provider to decide multiple options for transport services of interest, and use prices in the marketplace to indicate not only resource scarcity, but associated energy costs. Pre-informing the customers of various available choices allows customers to take the impact upon the network's current shape (as reflected by price) into account when exercising their choice *before* offering a traffic demand to the network. Thus we speak of *demand-grooming* or *choice-grooming*, rather than simple, after-the-fact, traffic grooming. By involving the customer, indirectly, in resource allocation decisions, we find that our approach is able to improve network performance and energy profile, while providing each customer service that is more satisfactory to them.

It has been recognized that power savings must be attempted at all levels of the Internet [118] and the Energy Consumption Rating (ECR) Initiative has published a specification on the energy assessment of network and telecom equipment[5]. The primary metric proposed is Energy Consumption Rating (ECR) in terms of the energy efficiency expressed in Watts/Gbps.

Estimates put the power expended by the Internet at nearly 10% of the total consumption in the USA already, poised to double in the next 10 years. The critical part of this power consumption is seen to be in server farms and routers, not at individual homes [136]. More seriously, the dissipation of heat is becoming a serious problem at the network switches and routers. For core networks, it has been established that optical processing is cheaper than electronic processing in terms of the power consumption. The possibility of using optics for packet routing has received significant research attention [90] [123]. However, in [175], Tucker argues that because the power consumed by optical buffers implemented by Fiber Delay Lines, the benefit of optical packet switching in terms of the power consumption is limited. This somewhat discouraging conclusion comes from the fact that the current optical techniques do not excel at traffic processing at packet granularity, at least from the power consumption point of view. Nevertheless, this argument does not prevent us from taking advantage of optical equipment that process tens of $10/40$Gbps wavelengths at a much lower energy cost comparing with routers. To achieve this goal, a rigorous formulation is needed that captures the energy consumption characteristics of equipment/functionalities at all involved layers.

The art and science of converging available technologies, electronic and optical, for the access and core, for network-wide benefit, has been known as *traffic grooming* in the optical networking research literature. This area generated a good deal of research in the past, which largely focused on optimizing the number of wavelengths, or OEO conversions, in keeping with the techno-commercial need of the time. This has given rise to the narrow definition of traffic grooming as the act of multiplexing sub-wavelength flows into wavelength channels. The literature on grooming has been surveyed variously [43, 93, 115, 69], and aspects of the research topic reviewed comprehensively [44]. The range of literature on grooming contained in the surveys and tutorials above shows it to be a broad area, focused on multi-layer approaches to traffic engineering and resource placement/optimization, characterized by explicit representations of constraints and opportunities specific to optical layer technologies. By and large, the "classical" traffic grooming problem tries to provide a comprehensive solution for routing traffic on the virtual topology, virtual topology design and routing and wavelength assignment (RWA). In general, the objective of grooming studies has been to minimize OEO (Opto-Electro-Optic) processing, which requires costly high speed electronics. In [70], our research group's previous work argued that grooming for OEO goals was a special case of traffic and network management under subwavelength traffic to optimize network goals, especially under dynamic traffic scenarios. In recent times, there have been various research studies that propose approaches for "green grooming" – grooming with the objective of optimizing network energy expenditure – including some of our own work [72].

In this chapter, we build on such works, but with a goal of making the optimization goals more customer-specific. We show that this approach allows each customer, individually, to be more satisfied with their service, while allowing the network to strike a satisfactory balance between satisfying customers and reducing energy costs. First, we show a network model that captures the main consumers of power. After that, we introduce two different ways to formulate the network node's power consumption, namely a flow based formulation and an interface based formulation. We then study present the results of a choice-based approach for green grooming to address the problem that increases the quality of service as well as the energy efficiency.

5.1.1 Network Node Model

An optical network can be abstracted as a digraph where each node is a set of network equipment responsible for handling traffic at different layers and each arc is an optical fiber. The power consumption of a network consists of the power consumption of network node and line equipment, mainly amplifiers. However, in practice the power consumption of amplifiers is usually negligible comparing with that of network equipment (e.g., a JDSU OA 400 EDFA consumes 4.5w), so in our model we omit the amplifiers' contribution to the overall power consumption and focus ourself on the power consumption of network nodes. Figure 5.1 shows a model for an optical network node whose functionalities can be classified into three layers, the optical layer, circuit layer and packet layer. Notice that each functional block may not have a one-to-one correspondence with real equipment. For example, the fiber switch may be an independent equipment that switches an incoming fiber at an input port to an outgoing fiber at an output port. Likewise, the ROADM and OXC are usually two independent equipment. Also notice, we assume the circuit layer equipment is able to handle SONET frames, however, if Ethernet packets are directly transmitted over DWDM, it can also be an Ethernet switch. The figure shows three possibilities how a traffic demand that is not sourced from or destined to this node can be processed:

1. the traffic demand bypasses this node optically.

2. the traffic demand is electronically processed by a switch at the circuit layer.

3. the traffic demand is electronically processed by a router at the packet layer.

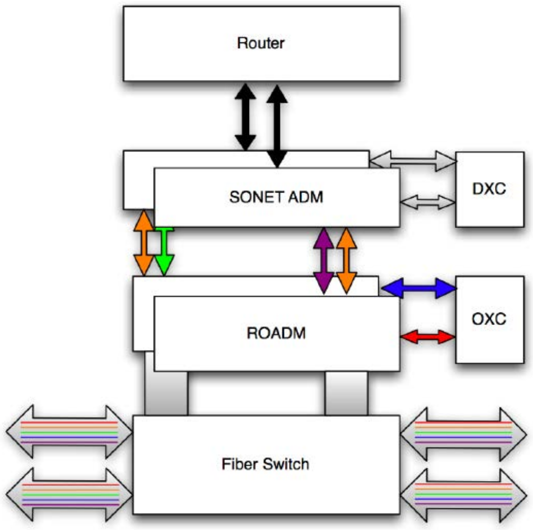


Figure 5.1: Model for an optical network node, taken from [71]

By distinguishing these three possibilities, we assume the 1st possibility is the most energy efficient and the 3rd possibility is the least energy efficient. This assumption is reasonable because it is shown in [14] that regeneration of traffic (OEO processing) consumes a large portion of the entire power consumption of a network.

5.1.2 Energy Expenditure Models

In this section, we propose a general view of modeling energy consumption of the network, and show how it can be used to reflect different network device characteristics, by providing two such formulations. The former is a flow based formulation that captures the potential of dynamic traffic-responsive energy savings by assuming the power consumption of equipment can be ideally described as a linear function of the traffic load. On the contrary, the latter approach captures the static aspect of energy consumption more prevalent today, by neglecting the impact of traffic load. As energy considerations in the network become critical, it can be reasonably expected that successive generations of network devices will improve their energy consumption characteristics, transitioning from the latter model to approach the latter, thus our general formulation is conceptually useful. It is also applicable to intermediate stages in that development.

General Formulation

We first present a detailed and somewhat idealized analysis of the power consumption utilizing the ECR metric. We distinguish three functionalities of network equipment: receiving, processing and transmitting traffic. Once the ECR of these three entities are known, then the power consumption of $\mathtt{t}$ amount of traffic traversing node $\mathtt{i}$ can be described. In order to present the general model, we introduce the following terminology. We indicate the rate at which energy is expended (i.e., the power expenditure) in a device or subsystem when it is used at full traffic load, normalized by the traffic load, by the "energy efficiency" of the device. Thus the energy efficiency of the device is the Watts/Gbps expended when operating at full load, and is equivalent to the ECR = $\mathsf{E}_{100}/\mathsf{T}_{\mathsf{f}}$ as defined in [5]. If the device is ideal, then at any intermediate load, the power expenditure is proportional to the load with this quantity. However, for practical devices, the power expenditure will be super-linear, and may have various characteristics as shown in Fig. 5.2. In particular, the power consumption may be a step function, such that the full rated power is consumed whenever the device is switched on. We formalize this notion below.

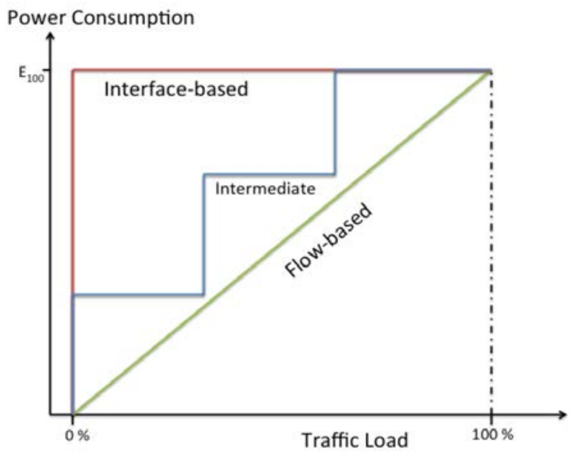


Figure 5.2: Power consumption characteristics

Optical bypassing

$$P_{n,t} = p_{oi}(t, e_{oi}) + p_{op}(t, e_{op}) + p_{oo}(t, e_{oo}), \tag{5.1}$$

where e_{oi} is the energy efficiency of the optical input port, e_{op} is the energy efficiency of the optical switching fabric, and e_{oo} is the energy efficiency of the optical output port. The functions $p_{oi}(\cdot,\cdot), p_{op}(\cdot,\cdot), p_{oo}(\cdot,\cdot)$ are *characteristic consumption functions*, that reflect the behavior of power expenditure as a function of traffic load, parameterized by its behavior at 100% load. Thus an ideal optical input port that consumes power linearly in the incoming traffic rate would be modeled by $p_{oi}(x, y) = xy$, while one with a constant drain would be modeled by $p_{oi}(x, y) = y$, the step function.

Above and in the rest of this section, we have used a notational simplification for the sake of convenience. Each characteristic consumption function and energy efficiency paramter should be further subscripted by n, to indicate that these are node-specific parameters; we have eliminated that subscript.

Circuit switching

$$\begin{aligned} P_{n,t} &= p_{oi}(t, e_{oi}) + p_{od}(t, e_{od}) + p_{si}(t, e_{si}) + p_{sp}(t, e_{sp}) \\ &\quad + p_{so}(t, e_{so}) + p_{oa}(t, e_{oa}) + p_{oo}(t, e_{oo}), \end{aligned} \tag{5.2}$$

where e_{od} is the energy efficiency of ‘drop’ port of the optical switch such as ROADM, e_{si} is the energy efficiency of the input port of the frame switch such as a SONET switch, e_{sp} is the energy efficiency of the SONET switching fabric, e_{so} is the energy efficiency of the SONET switch’s output port, and e_{oa} is the energy efficiency of the ‘add’ port of the ROADM. The various $p(\cdot,\cdot)$ functions are characteristics functions of the various corresponding devices and subsystems as above. In this equation, we distinguish the energy efficiency of an optical input/output port and that of a local ‘add/drop’ port because in practice, short reach transceivers are often used for local ports to save the cost. The difference between the previous equation and this equation is that the $p_{op}(t, e_{op})$ term that represents power consumed by optical processing is replaced by the combined power consumption of the optical add/drop process and the electronic frame switch, which is typically significantly larger.

Packet processing

$$\begin{aligned} P_{n,t} &= p_{oi}(t, e_{oi}) + p_{od}(t, e_{od}) + p_{si}(t, e_{si}) + p_{sd}(t, e_{sd}) \\ &\quad + p_{fi}(t, e_{ri}) + p_{rp}(t, e_{rp}) + p_{ro}(t, e_{ro}) + p_{sa}(t, e_{sa}) \\ &\quad + p_{so}(t, e_{so}) + p_{oa}(t, e_{oa}) + p_{oo}(t, e_{oo}), \end{aligned} \tag{5.3}$$

where e_{sd} is the energy efficiency of the SONET switch's output port that is connected to a router, e_{ri} is the energy efficiency of the router's input interface, e_{rp} is the energy efficiency of the router's packet processing, e_{ro} is the energy efficiency of the router's output interface, and e_{sa} is the energy efficiency of the SONET switch's input port that is connected to the router. In this equation, we show that the $p_{sp}(t, e_{sp})$ term that represents the power consumed by SONET frame switching fabric is replaced, in turn, by the combined power requirements of Layer 3 add/drop from Layer 2, and of the store-and-forward action of packet forwarding, which is, again, significantly larger.

Flow based formulation

To obtain the flow-based formulation, as a first step, we simply replace each $p(\cdot, \cdot)$ function by the linear function, as discussed above. The resulting equations present idealized formulations of the power consumption by decomposing the equipment into major functional blocks. However, in practice, the power consumption of the input port/output port/switching fabric is implementation dependent. For example, a line card may integrate multiple input/output ports so that the ports can share the peripheral circuitry. Some implementation even integrates a local switching fabric on the line card to reduce the burden of the back plane. Therefore, it is usually unpractical to model the power consumption at a this level of details. In what follows, we present a formulation based on the traffic flow and the energy efficiency at a equipment level. Let the energy efficiency of the optical equipment, the circuit equipment and the packet equipment be e_o, e_s and e_p respectively. Let $t^n_{o,sd}$ be the traffic sourced from s, destined to d and optically bypassing node n, $t^n_{s,sd}$ be the traffic that is circuit switched at node i, $t^n_{r,sd}$ be the traffic that is packet routed at node n. Then the objective function is:

$$\min \sum_{n \neq s,d} \sum_{s,d} \left(t^n_{o,sd} e_o + t^n_{s,sd} e_s + t^n_{r,sd} e_r \right). \tag{5.4}$$

Notice that in this formulation, we omit the power consumption caused by traffic that is sourced from or destined to the node. Since the traffic is always dropped to local ports and sent to a local network such as an access network, we can assume it consumes a constant amount of power that can not be saved by a less power-consuming bypassing technique. For static grooming network, the traffic matrix is static and known a priori, we have $t^n_{o,sd} + t^n_{s,sd} + t^n_{r,sd} = t^n_{sd}$, which is the traffic demand from s to d traversing node n. Since $e_o << e_s << e_r$, Equation (5.4) is equivalent to

$$\min \sum_{n \neq s,d} \sum_{s,d} \left(t^n_{s,sd} e_s + t^n_{r,sd} e_r \right), \tag{5.5}$$

which encourages the strategy of minimizing the electronic processing of traffic demand.

In this section and the next, we have attempted a modeling of power consumption that provides conceptual value, not a complete formulation that solves the entire problem including the design of virtual topology formed by lightpaths, routing traffic on the virtual topology, routing of lightpaths and wavelength assignment With any specific power modeling, such a formulation is straightforward to achieve by utilizing standard existing traffic grooming formulation; for such a formulation, please refer to [70].

Interface based formulation

The flow based formulation forms an insightful if unrealistic guide to designing an energy efficient optical network. This formulation is nevertheless practically interesting because we anticipate more and more network equipment will be tested for its energy efficiency by ECR, and manufacture will be targeted to bring their power consumption closer to the linear model. However, in the short term, the energy efficiency defined as power consumption normalized to the effective throughput is inadequate for more detailed and accurate analysis. Moreover, it does not explicitly represent the opportunity presented to the designer by the fact that the energy efficiency of various network devices are widely different. As a simple example, suppose ten 100Mbps traffic demands are packet processed by a router, spreading the traffic load into ten router interfaces consumes much more power than consolidating them into a single 1-Gbps interface. The main reason for this is that the power consumption of network equipment is usually not a linear function of the load. For instance, for a single-chassis router, the power consumption is a linear function of the number of active interfaces if all interfaces are of the same type and the impact of traffic load is omitted [26]. Generally speaking, the power consumption of a network node can be decomposed into the static part and the dynamic part, denoted as p_s and p_d respectively. p_s is constant if it is powered on and p_d is configuration/load dependent. This model applies recursively into each network equipment and each line card. Applying this model to a node n that has L line-terminating equipment and each equipment i has K_i line cards, we have:

$$P_n = p_s + \sum_{i=1}^{L} \left(p_{es}^{i} + \sum_{j=1}^{K_i} \left(p_{ls}^{ij} + p_{rp}^{ij}(t_{ij} e_{ij}) \right) \right), \tag{5.6}$$

where P_n is the power consumption of the node, p_s is the power consumption of its static part, p_{es}^{i} is the static part of the power consumption of equipment i, p_{ls}^{ij} is the static part of the power consumption of line card j in equipment i, and $p_{rp}^{ij}()$ is the characteristic function for its dynamic part, which is some function of the traffic load t_{ij} and the energy efficiency e_{ij} of line card j of equipment i. Notice that this model is meant simply to illustrate its recursive nature instead of being comprehensive. For example, some network equipment may have multiple chassis; in this case, an extra level of summation over all chassis needs to be inserted into the equation.

This model implies that if the static power consumption of an entity is much larger than the dynamic part consumed by its components, it is more meaningful to minimize the number of the entities while having possibly a larger number of components. Using o, s, r to denote the optical equipment, SONET equipment, and router respectively, Equation (5.6) can be further expanded to represent the power consumption of these equipment separately, which we omit for brevity.

It has been observed that the static power consumption of a line card dominates the load dependent part [26]. If we further assume that no node is completely switched off, nor any LTE of any node completely switched off, switching off individual line-cards becomes the only avenue for energy minimization; and we can consider a further simplified interface based formulation that omits the impact of traffic load, with an objective to minimize network-wide power consumption:

$$\min \sum_{n=1}^{N} \left(\sum_{j=1}^{k_{n,s}} p_{n,ls}^{sj} + \sum_{j=1}^{k_{n,r}} p_{n,ls}^{rj} \right), \tag{5.7}$$

where N is the number of nodes in the network, and each node n has $k_{n,s}$ line cards connected to upstream neighbor nodes and $k_{n,r}$ line cards connected to downstream neighbor nodes. (We have added back the subscript indicating the node, since this expression spans multiple nodes.) If we assume the network is a SONET network, this problem is equivalent (since each node only has two types of equipment: ROADMs and SONET ADMs) to the well studied traffic grooming problem that aims at minimizing the number of SONET ADMs. This also underscores the fact that traditional traffic grooming is not necessarily green grooming except under a number of assumptions. Most of these assumptions are realistic at present, but energy characteristics of network devices are likely to evolve in the near future; hence our interest in the general formulations.

5.2 Choice Based Cooperative Optimization

We make a few basic observations which guides us to take a novel approach to the combined problem of allowing the future backbone network of tomorrow to exercise more energy-efficient technology as it becomes available, while not reducing the satisfaction level of customers in general.

- As mobile clients as well as short-lived virtual machine clients of the network become more powerful, with increasing access speeds, cloud computing and even some e-science applications create shorter but significantly heavy data flows, creating increasingly dynamic traffic demand patterns, especially at the access level, but also at higher levels of the aggregation hierarchy.

- The optimization space for an energy-efficient network extends across multiple layers, and across packet/virtual-circuit/optical-circuit networks, as we have explicitly seen above. As a consequence, the lightpath network cannot be protected completely from the rapid dynamics of the packet layer, if it is to be responsive to the needs of energy optimization.

- Different customers for the same network provider have different requirements, and different levels of tolerance for degradation of service for optimization of the whole network.

- Customers in general can be depended upon to know their own priorities, and attempt to maximize their benefit, if they have access to mechanisms allowing them to act on their knowledge. Similarly providers can be depended upon to know the relative costs, capex and opex (including energy and cooling), of operating their network devices and resources, and can be depended upon to do so economically as long as their revenue and profit is not affected.

- Control planes for packet, frame and lightpath networks are varied, complicated, and difficult to extend or standardize. Any extensions in signaling or auto-negotiation for enhancing the energy-efficiency of backbones would have to go across layers, and would be difficult at best.

We envision, in the light of our ongoing ChoiceNet project discussed in Chapter 3, that a service marketplace that acts as a rendezvous between provider and customer can act as a mechanism for cooperative optimization of network resources in an agile optical backbone of the future, such as path or pathlet services, or even in-network processing services. However, in the ChoiceNet view, the customer would undertake individual contracts with each of the providers of such services along the complete path to compose the entire service, and would be financially rewarding each provider, thus creating incentive for innovation at all such service points, endpoints, paths, and intermediate processing/storage services. In reality, automated software acting on behalf of the user would undertake these decisions, translating high-level goals of user experience into low-level ones.

5.2.1 Choice as Optimization Mechanism

Not every customer has the same requirement, thus a network control strategy that embodies the whole network is bound to satisfy some customers less than others. At the same time, since network decisions like path selection and OEO/OOO decision at intermediate nodes are inextricably coupled for multiple customers, performance optimization can only be done for the network as a whole. Moreover, the desirability of such decisions from the network point of view (say, the energy penalty for a a given path selection and OEO/OOO decision for a given flow) changes every time the network traffic occupancy changes; what is a green decision at one point may carry a high energy penalty at another. The provider is in a position to track such changes but individual customers are not.

We propose that *choice* is the key to converging the needs of the customer and the network. Our hypothesis is:

Cooperative Optimization Hypothesis If providers provide multiple alternative services in response to individual customer requirements and advertise costs for the alternatives that correctly reflect their capital and operating cost (including energy), and customers each individually make their own choices for service by responding to their individual priorities, then the entire network will be better optimized with respect to conflicting performance goals, leading to a higher degree of collective customer satisfaction, than if the network attempts to make specific service selections on the basis of any single performance metric for all customers.

In the next section, we study this hypothesis, under a simplified abstraction of conflicting network performance goals. Specifically, we study the conflict of minimizing the energy expenditure and minimizing the latency of new traffic demands. We do not attempt to represent the full economy plane or signaling requirements of ChoiceNet, but simply assume the existence of a marketplace in which providers can advertise single or multiple alternatives for a point-to-point demand, as shown in Fig. 5.3. We make no assumptions or requirements regarding when the provider computes these alternatives; they may be done entirely on-demand, when the point-to-point traffic demand arrives at the source node, or the provider may decide to pre-compute, on an ongoing basis, alternatives for the most common or expected traffic demands. An alternative for a specific demand may utilize any combination of available packet and optical resources available in the network at the time. As a first step, we assume that the provider computes the lowest-delay alternative and the lowest-energy alternative. The advertisement of each alternative includes its latency, and a price. The price is derived from a combination of the equipment cost and energy cost to the provider.

5.3 Demand-Grooming Study

In our study, we do not show details of computation of the lowest-latency and lowest-energy alternatives that the provider must perform. The computation is straightforward,

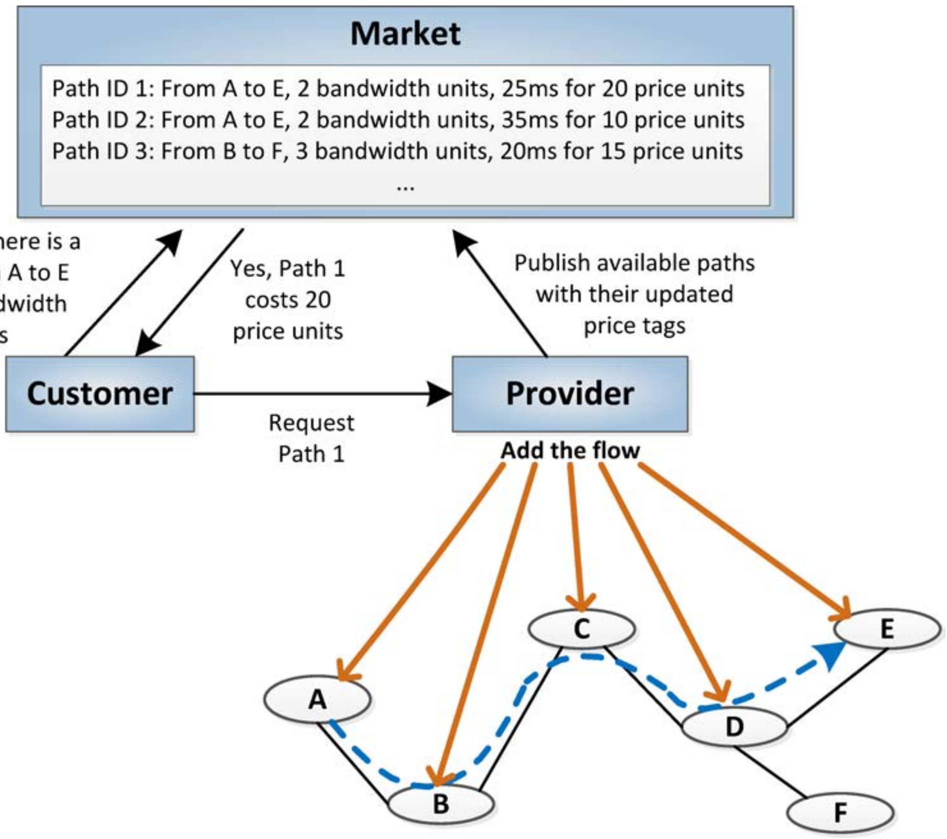

Market
Path ID 1: From A to E, 2 bandwidth units, 25ms for 20 price units
Path ID 2: From A to E, 2 bandwidth units, 35ms for 10 price units
Path ID 3: From B to F, 3 bandwidth units, 20ms for 15 price units
...
Checks if there is a path from A to E for 2 bandwidth units
Yes, Path 1 costs 20 price units
Publish available paths with their updated price tags
Customer
Provider
Request Path 1
Add the flow
A
B
C
D
E
F

Figure 5.3: Market Approach

by maintaining a capacitated network connectivity layered graph, similar to [193] and many other previous work. In this graph, each network node is split into multiple graph vertices that represent the various stages of the node (wavelength, subwavelength etc.) and all possible OEO and OOO paths through the network are available to a shortest-path algorithm. However, when a path is provisioned for an arriving demand that precludes certain other configurations of the switch, the arcs corresponding to those configurations are assigned capacities of zero until the existing traffic departs. For example, if a wavelength is passed through a node from a particular input port to a particular output port to form a lightpath, then the arcs from the vertex representing the digital switch to the vertices representing the input and output port for that wavelength are each assigned a capacity of zero, indicating that the wavelength cannot be used for OEO traffic injection or extraction any more. Then the computation of the alternatives is a simple matter of maintaining two sets of arc weights, one representing latency and one representing energy cost, and running Dijkstra's algorithm once on each set. We refer to these as the FAST and GREEN alternatives. Fig. 5.4 shows an example of such paths. In what follows, for any point-to-point arriving demand, we denote the delay and energy cost of the FAST path by $\mathsf{d_f}$ and $\mathsf{e_f}$ respectively, and the delay and energy costs of the GREEN path by $\mathsf{d_g}$ and $\mathsf{e_g}$ respectively.

5.3.1 Simulation Model

In order to compare the effects of different choice selections, we have created five different customer behaviours as follows;

1. **FAST:** The arriving customer always chooses the FAST path (i.e. the alternative with minimum delay cost) without considering the energy costs.

2. **GREEN:** The arriving customer always chooses the GREEN path (i.e. the alternative with minimum energy cost) without considering the delay costs.

3. **DELAY PREF:** The customer compares the advertised delay costs of the FAST and GREEN paths. If $\mathsf{d_g}$ exceeds $\mathsf{d_f}$ by a factor of α or more, then FAST path is chosen, otherwise GREEN path is chosen. The α value depends on the customer policy.

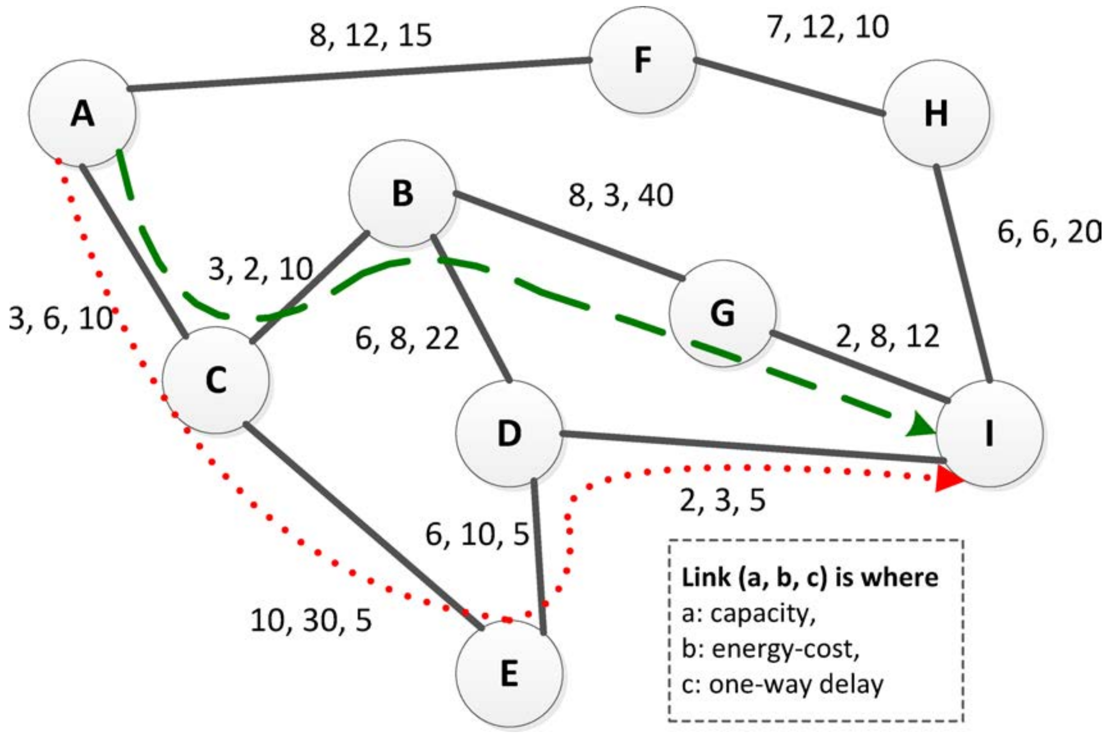


Figure 5.4: An example showing the FAST vs GREEN paths

4. **ENERGY PREF:** The customer compares the advertised energy costs of FAST and GREEN paths. If e_f exceeds e_g by a factor of β or more, then GREEN path is chosen, otherwise FAST path is chosen. The β value depends on the customer policy.

5. **HALF:** The arriving customer chooses FAST or GREEN randomly each with 50% probability.

In the simulations below, we use the customer models as follows. A simulation run in which all customers are made to behave as FAST customers is equivalent to a network provider which does not give customers any choice, but assumes that all customers always want the minimum latency option. Similarly a simulation run with all customers forced to be GREEN is equivalent to the case of a provider which assumes all customer demands can be routed with a view to minimizing total network energy expenditure (the algorithms we presented in Section 5.1.2 represent such provider behavior). Thus, both of these scenarios refer to the traditional non-choice-based customer-provider interaction.

In contrast, the last three scenarios represent choice-based scenarios. The HALF scenario simply assumes that each customer is a determined FAST or GREEN customer, but the entire population of customers is evenly divided between them. The other two represent customer populations which have a specific inclination to be either FAST or GREEN, but are not completely oblivious to the conflicting metric; thus they are somewhat responsive to how current network resource availability is affected by their choice.

The simulation is implemented in C++ [117]. In the simulation, when a customer request arrives, if there is available resource for the request in the network, the customer's flow is added to the network and stays in the network until its holding time expires, otherwise the request is blocked and cleared. Table 5.1 shows the specific values we have used to represent the component energy costs and other parameters of the simulation. We do not represent customer defection; we assume that an arriving request will result in admitting a flow into the network (in accordance with customer policy) unless it is blocked. We also make the following simplifying but reasonable specific assumptions to drive the simulation:

- For each wavelength used, there is an energy cost associated with the laser to transmit and receive optical signals. We combine them as the wavelength cost. We set it fixed and same for all wavelengths at all nodes.

- The electronic switching and routing are represented as a single OEO (opto-electronic-opto) operation and its energy cost is multiplied by the traffic volume passed over this domain.
- Each edge is bi-directional and each direction has 10 wavelengths, where each wavelength has a capacity of 10 bandwidth units.
- Customer requests arrive with poisson distribution and their service times are exponentially distributed.
- Each customer request can be from any node to any node with uniform distribution. In addition, each request can be either 1, 2, 5 or 10 bandwidth units each with 25% probability.
- Each experiment involves four runs of the simulation with differing arrival rates (contributing to traffic load), representing low traffic to higher traffic cases. In addition, each such experiment is repeated 30 times with different set of randomly generated customer requests. All results are calculated with their respective 95% CI (confidence intervals).
- Assigned delay costs are approximated based on the propagation distances.
- The network never rearranges currently resident traffic. Each demand retains the initially assigned path until it departs.

We have run the same simulations on two commonly used graphs; NSFNET (14 nodes and 21 edges) and USNET (24 nodes and 43 edges) [101] network topologies. Our results for both are very similar; for brevity we present only one set.

5.3.2 Metrics

Let $\mathfrak{b}_i$, $\mathfrak{d}_i$ and $\mathfrak{e}_i$ be the i^{th} accepted customer flow's bandwidth units, delay and energy cost respectively.

Weighted Delay is the average delay of all admitted customer requests, weighted by their respective bandwidth units.

$$WeightedDelay = \frac{\sum_i d_i b_i}{\sum_i b_i} \tag{5.8}$$

Weighted Energy Cost is the average delay of customer requests weighted by its bandwidth units.

$$WeightedEnergy = \frac{\sum_i e_i b_i}{\sum_i b_i} \tag{5.9}$$

Block Ratio is the percentage of total bandwidth units of accepted customer requests over the total bandwidth units of all offered customer requests. Let b_t be the t^th offered customer request's bandwidth units, then

$$BlockRatio = \frac{100 \sum_i b_i}{\sum_t b_t} \tag{5.10}$$

Total Utilization is the average percentage of network's usage. Let ${hc_i}^t$ represent the number hops (edges) used by i^th flow at time t and let e, w, c, T represent the number of directed edges in the network, the wavelengths per edge, bandwidth unit capacity per wavelength and simulation time respectively.

$$UtilizationRatio = \frac{100 \sum_t^T \sum_i hc_i^t}{ewcT} \tag{5.11}$$

Total delay and **total energy** cost refer to the sum of all delays, and sum of all energy costs weighted by their bandwidth units respectively.

$$TotalDelay = \sum_i d_i b_i, TotalEnergy = \sum_i e_i b_i \tag{5.12}$$

5.3.3 Numerical Results

Figure 5.5-5.11 show the value of the above metrics for the USNET set of simulations. Confidence intervals of 95% have been drawn for each data point, but on most it is too small to be discernible. As expected, the average delay is least for the non-choice-

Table 5.1: Simulation Configuration Parameters

Parameter	Values
Simulation Time Per Case	300 minutes
Average Holding Time Per Request	30 minutes
Requests Per Minute Per Case	4, 8, 12, 16
α (Delay Preference Ratio)	%20
β (Energy Preference Ratio)	%60
OEO Energy Cost	20w per bandwidth unit
Wavelength TX and RX Energy Cost	20w
Optical Switching Energy Cost	1w
ADD/DROP ROADM Energy Cost	4w
Router in/out Energy Cost	1w
OEO Delay	1ms
ADD/DROP ROADM Delay	1ms
Router in/out Delay	1ms
Optical Switching Delay	1ms

based case where all customers are treated as FAST customers, but it carries the highest energy penalty. Conversely, the non-choice-based case where all customers are treated as GREEN naturally has the best (lowest) energy expenditure, but the highest delay profile. This is obvious since in many cases an arriving point-to-point demand can be easily groomed into a longer path by utilizing existing lightpaths, thus reducing energy cost, in the GREEN case. But in the FAST case, each arriving demand will be groomed into its least latency path, even if it involves going through more OEO, or setting up new lightpaths, imposing a high energy penalty.

In contrast, the choice-based approaches are able to obtain reasonable, medium values for each metric without imposing a heavy penalty in the other. As expected, the DELAY PREF curve tends to be close to the FAST one, and the ENERGY PREF tends to the GREEN one. The ENERGY PREF and HALF cases perform in between FAST and GREEN. For the weighted energy figure, ENERGY PREF and GREEN perform significantly better for all request rates. The Block ratio and Network utilization are similar for all choices, but GREEN choice has slightly more Block ratio and utilization. This is due to the fact that GREEN choice tries to prevent the expensive OEO operation by finding different (and longer) paths so it uses more wavelength resources and gets blocked quicker than others. On the other hand, ENERGY PREF choice consistently has lower

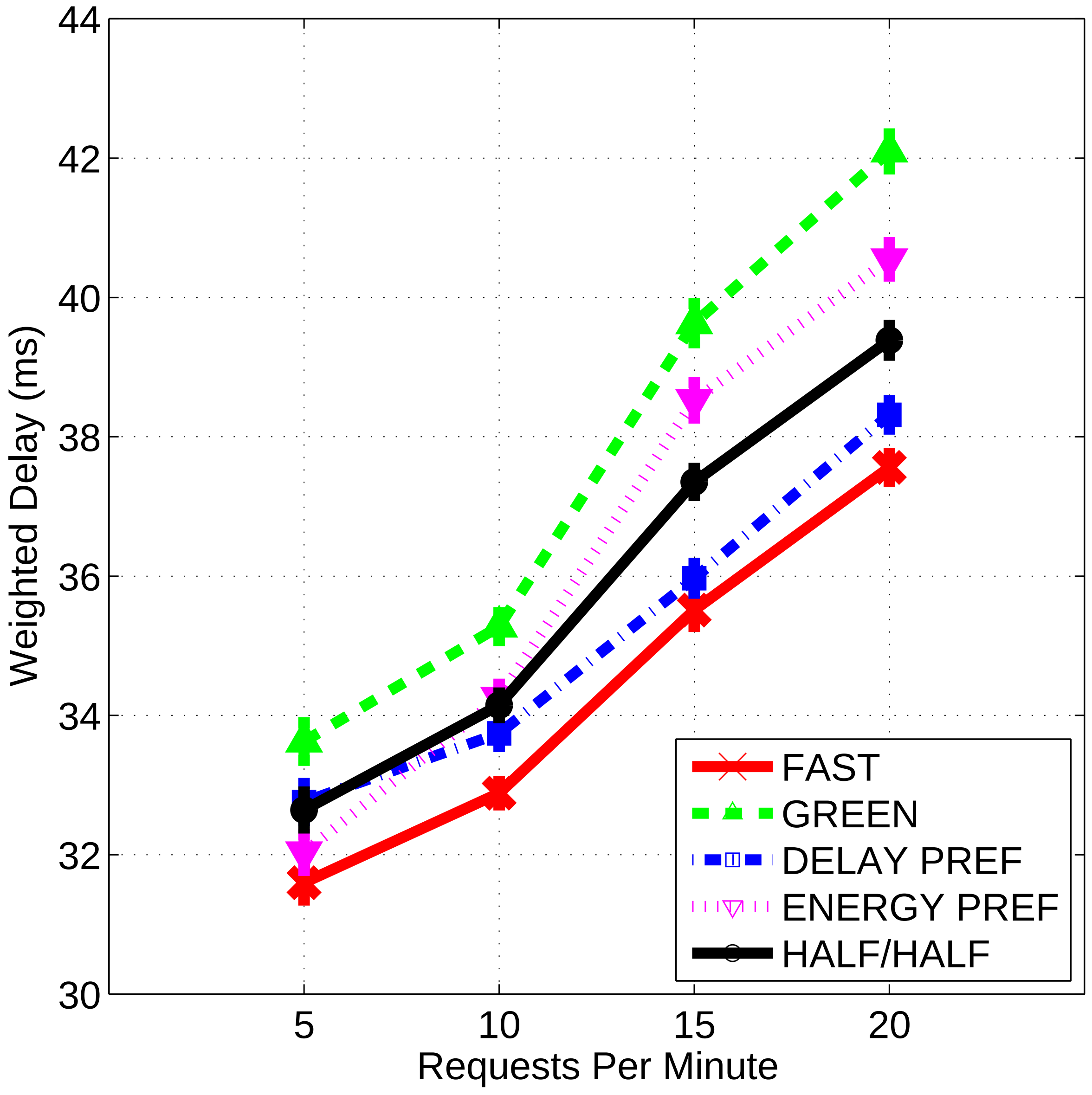


Figure 5.5: Weighted Delay

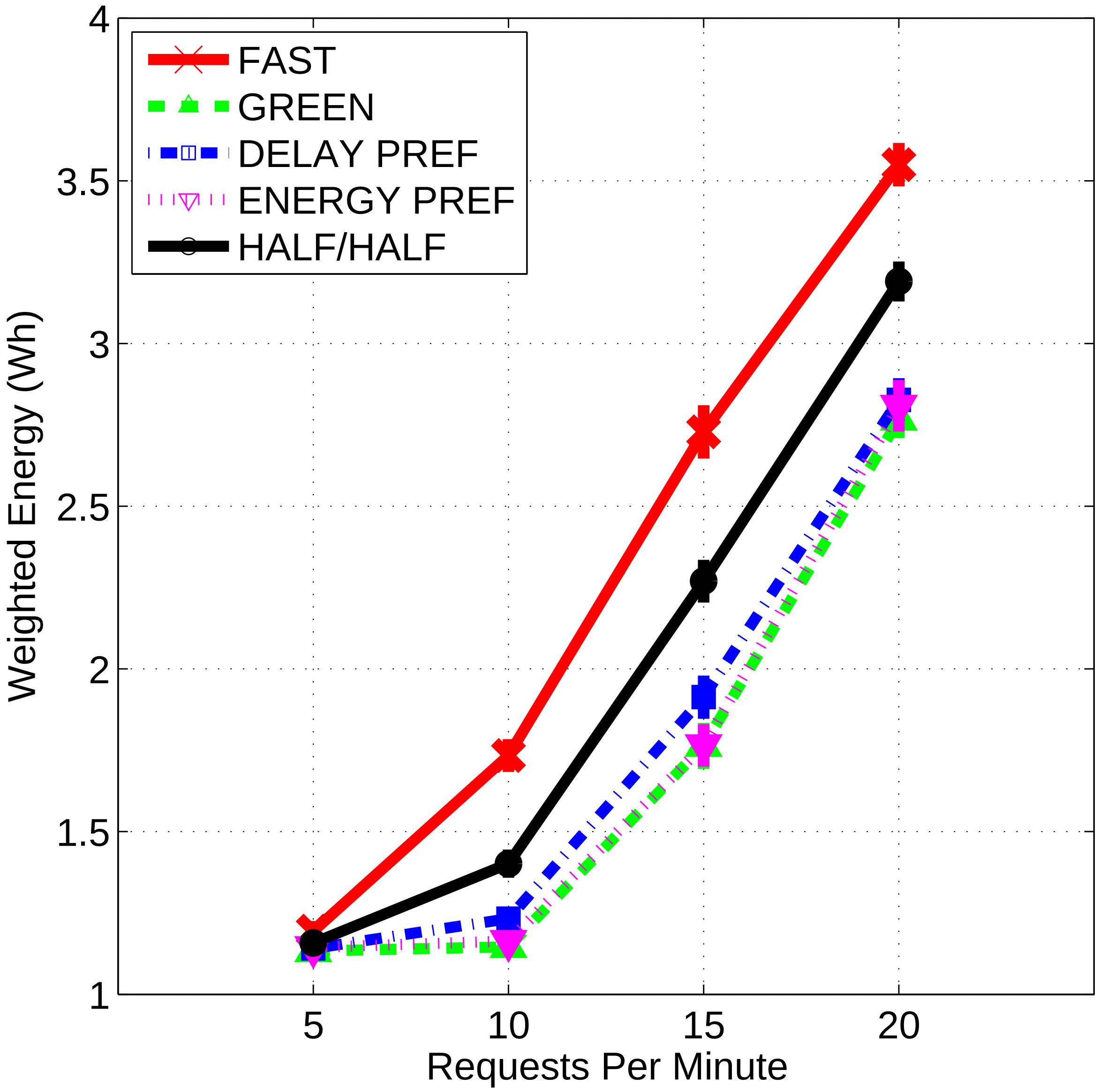


Figure 5.6: Weighted Energy

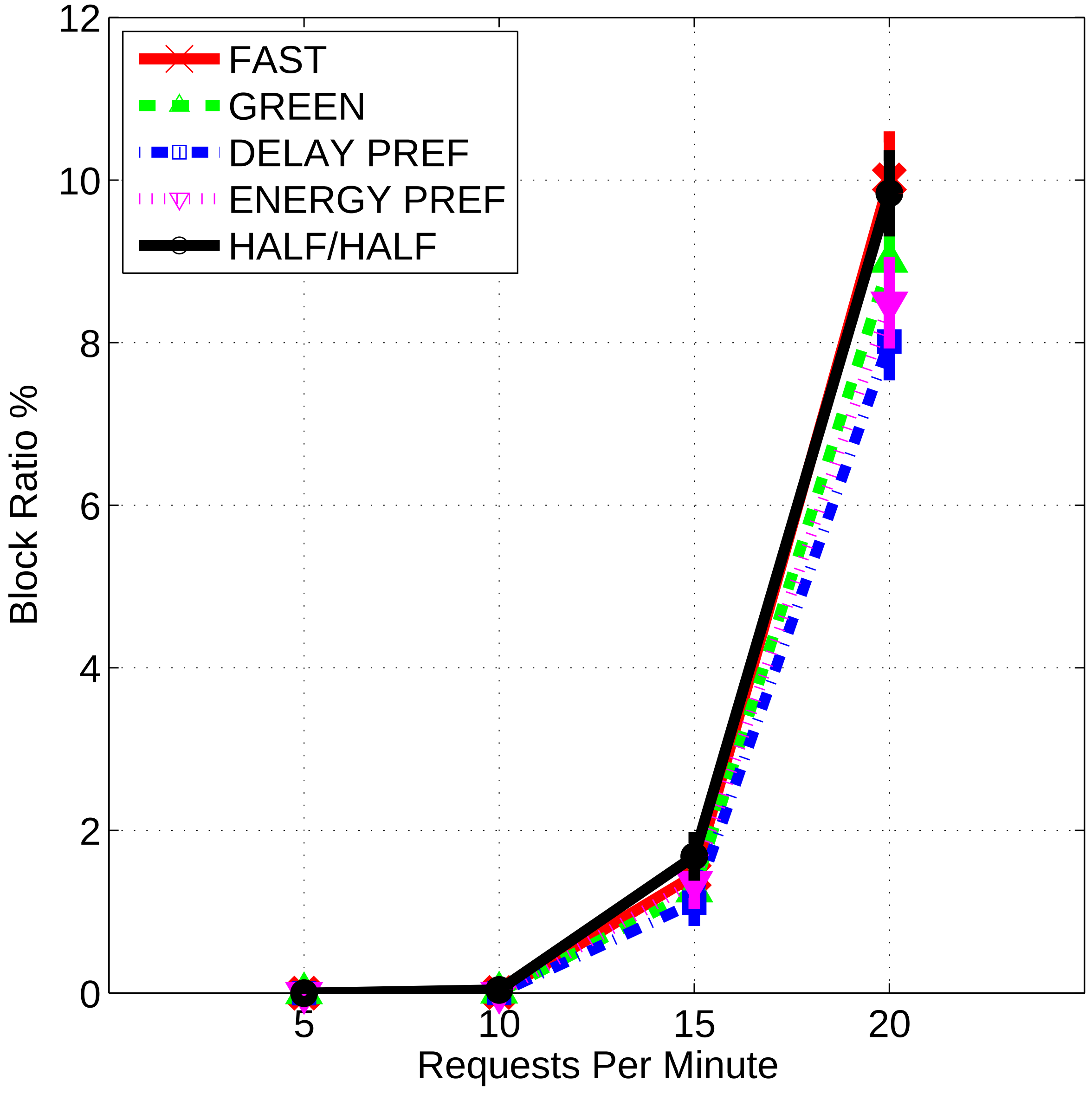


Figure 5.7: Block Rate

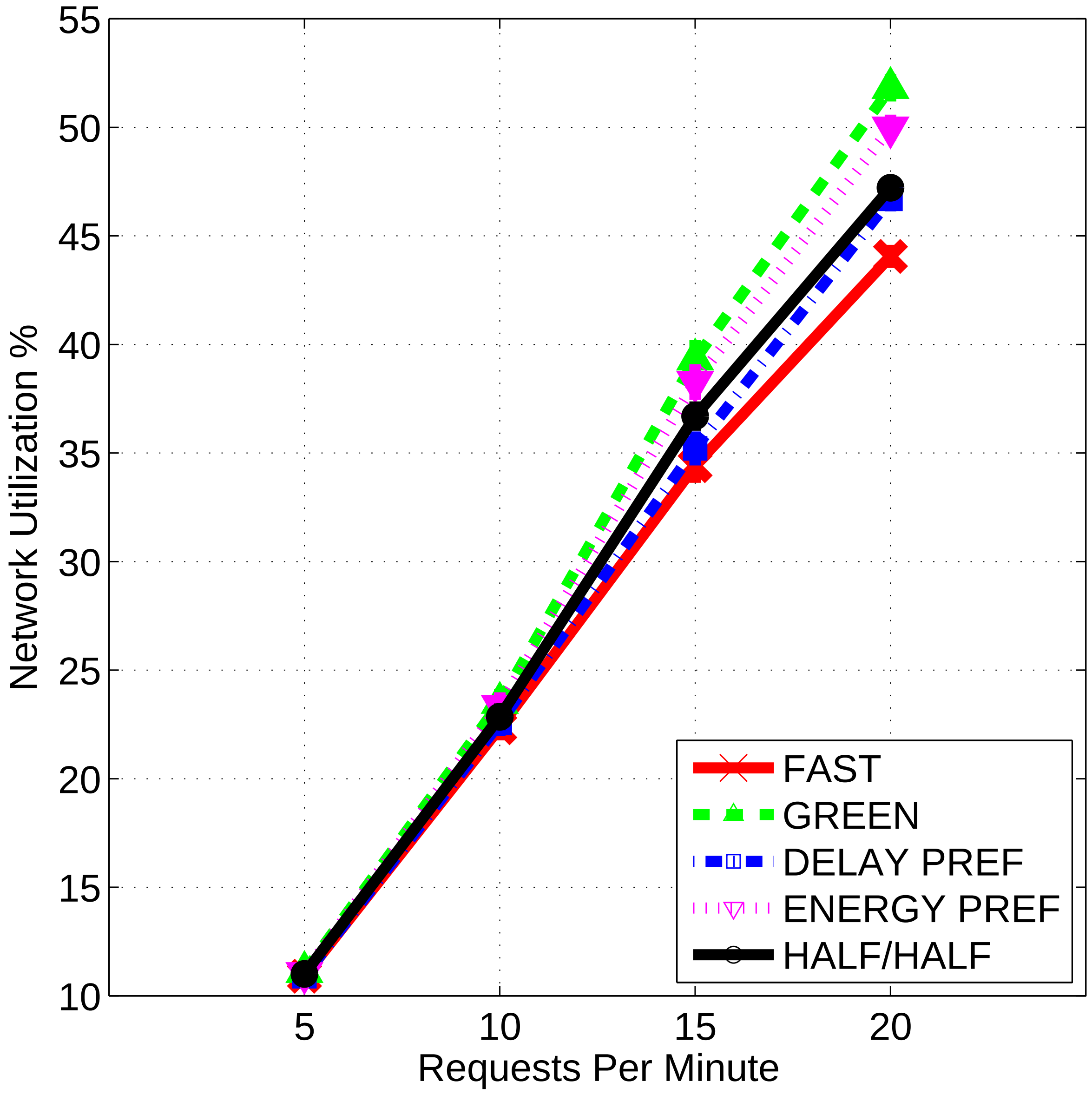


Figure 5.8: Utilization

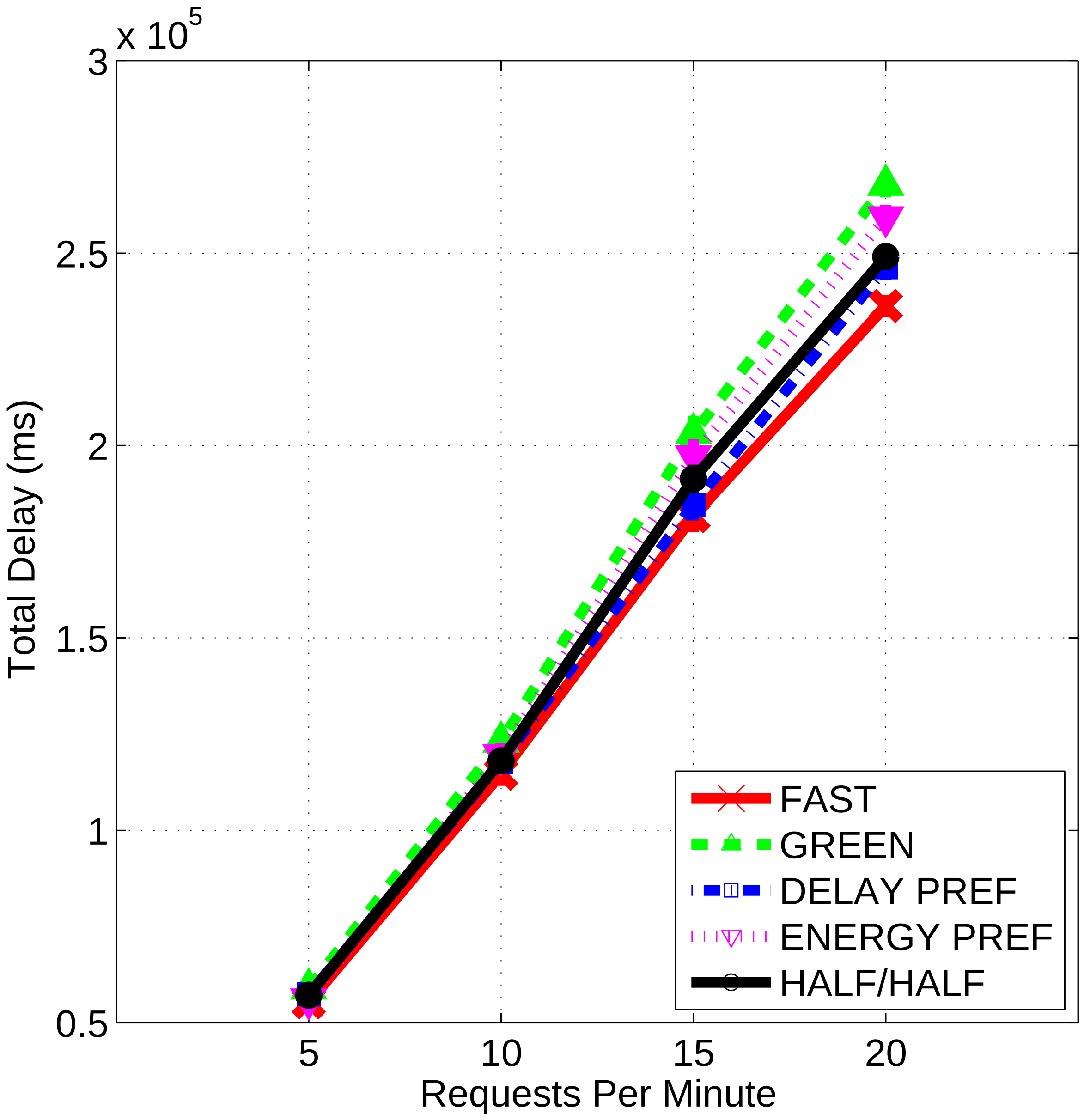


Figure 5.9: Total Delay

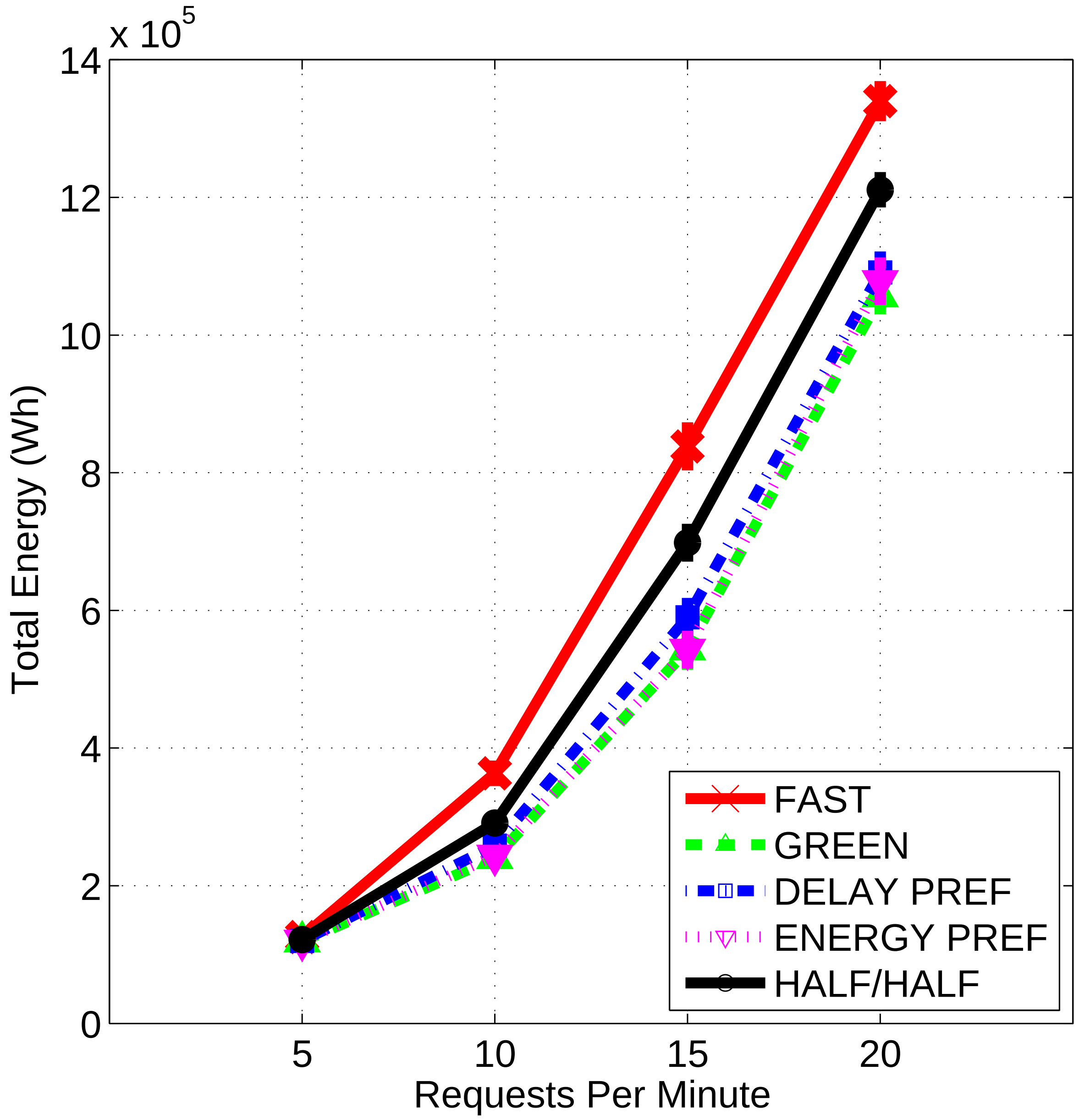


Figure 5.10: Total Energy

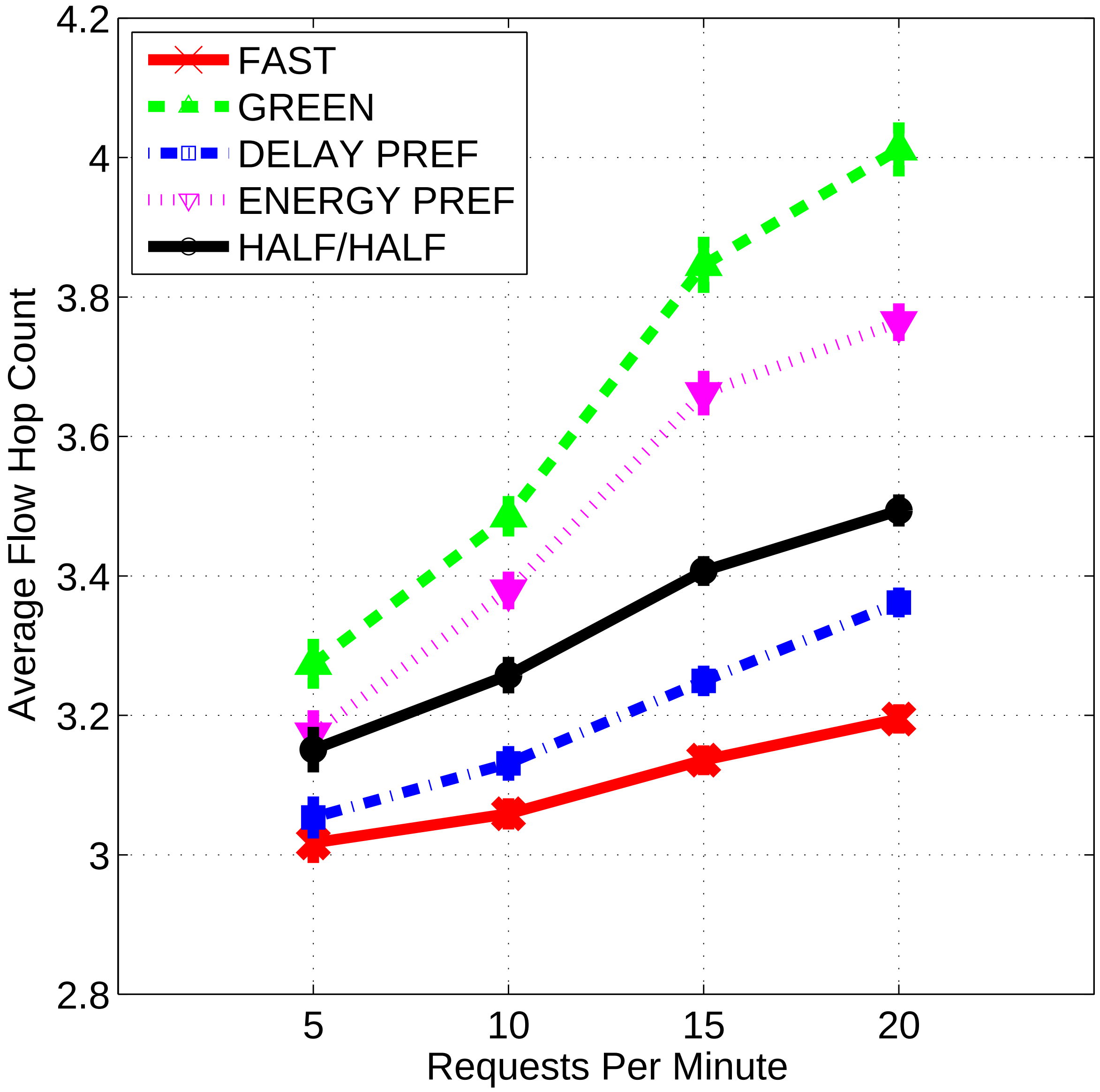


Figure 5.11: Hop Count

hop count, block ratio. This is due to the fact it prefers FAST path if the energy difference is less than 60%. We also observe that ENERGY PREF performs better than GREEN at some point. This is because GREEN choice case greedily uses cheap paths and saturates the network quickly, so all remaining paths are costly. This is further confirmation that the greedy approach is far from optimal in satisfying customer demands, as we have also seen in [71].

Overall, both the ENERGY PREF and DELAY PREF choices show significantly more energy savings than the FAST choice, while at the same time providing much better quality service then the GREEN choice. For example, Figures 5.5 and 5.6, at 15 Request per Minute, DELAY PREF has an average of 36ms whereas GREEN has 39ms; around 10% increase. On the other hand, its energy cost is almost same as the GREEN choice. This gives a clear win-win scenario, where the provider is using less energy and therefore less cost but at the same time the customers are receiving less delay.

While this appears to address the network provider's needs, we now examine the satisfaction customers can expect from their service by taking a closer look at the HALF case. Fig. 5.12-5.17 present the same metrics as above for the HALF case only, but now we also observe the values of those metrics as experienced by only the FAST customers and only the GREEN customers among the total customer population separately. Fig. 5.12 and 5.17 clearly show that the choice-based approach is able to maximize the satisfaction of the customers – FAST customers receive low latency services, and GREEN customers receive low energy services. In fact these metrics are, as perceived by respective customer populations, at least as good as if the entire network had aggressively optimized for the corresponding metric, as can be verified by direct comparison with Fig. 5.5 and 5.6. Further, Fig. 5.14 and 5.15 shows that the network is able to provide this differentiation in service without discriminating between the two populations in terms of admission into the network or share of network utilization. It is also worth keeping in mind that with the cooperation of the customer, the provider is able to achieve this without computing complicated optimization algorithms dynamically, merely by computing two distinct choices for each request.

In the above, we have taken reasonable values of the parameters from the literature and available device configurations from manufacturer specs, but we are well aware that the results may not reflect specific other network conditions. However, our conclusion is not quantitative in nature, but rather we view the above as a proof of the concept of

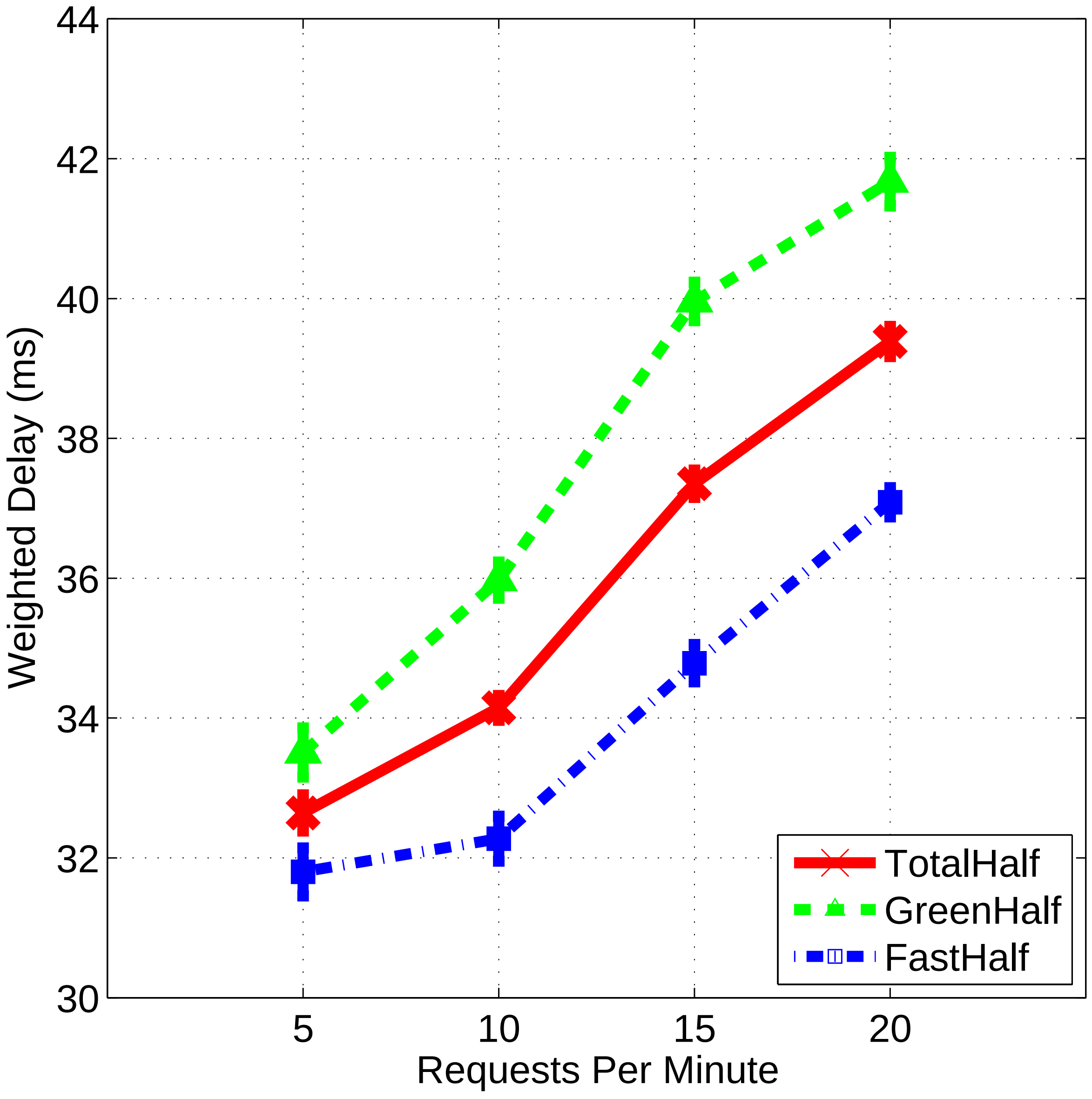


Figure 5.12: HALF case detail: Weighted Delay

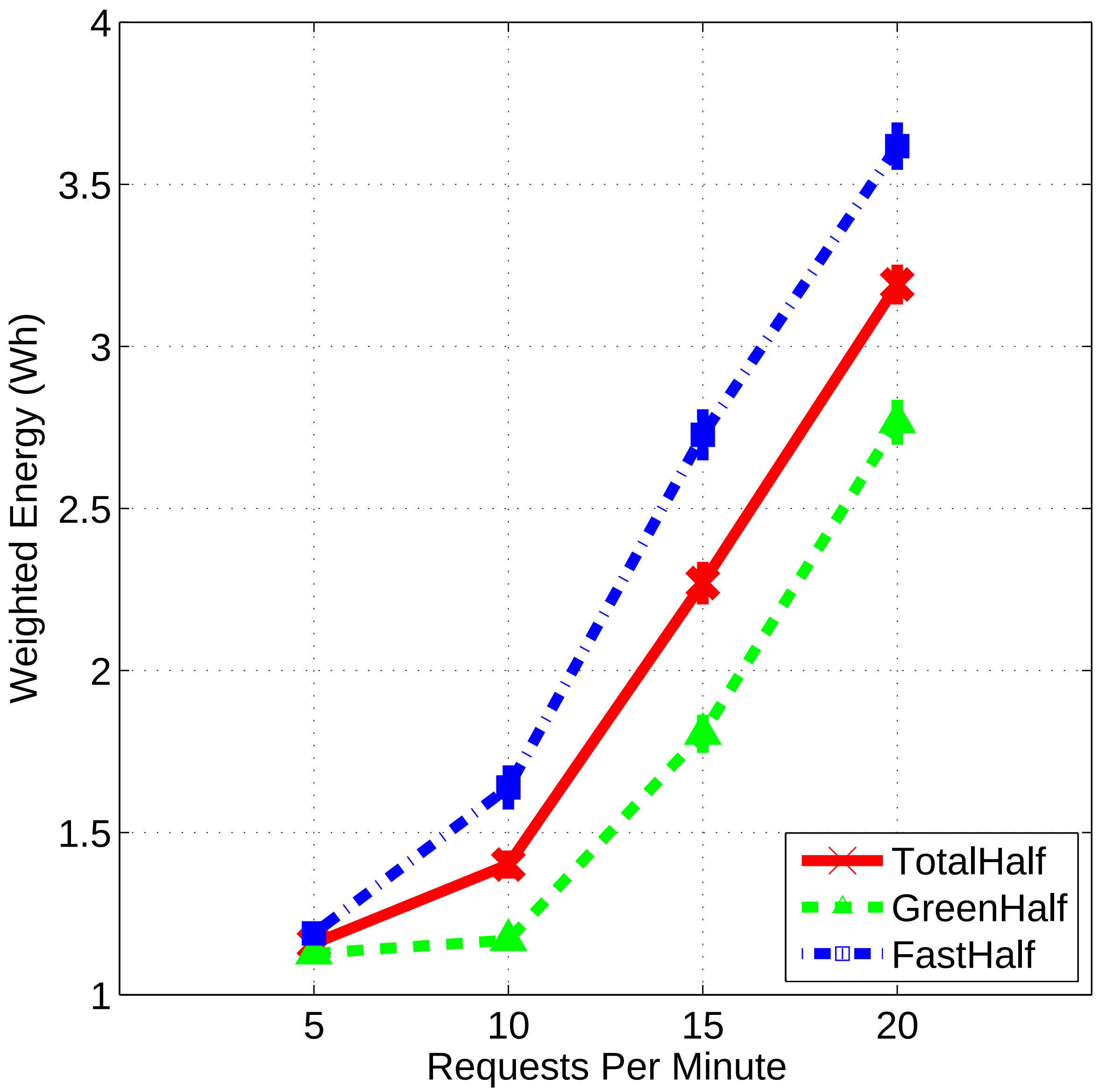


Figure 5.13: HALF case detail: Weighted Energy

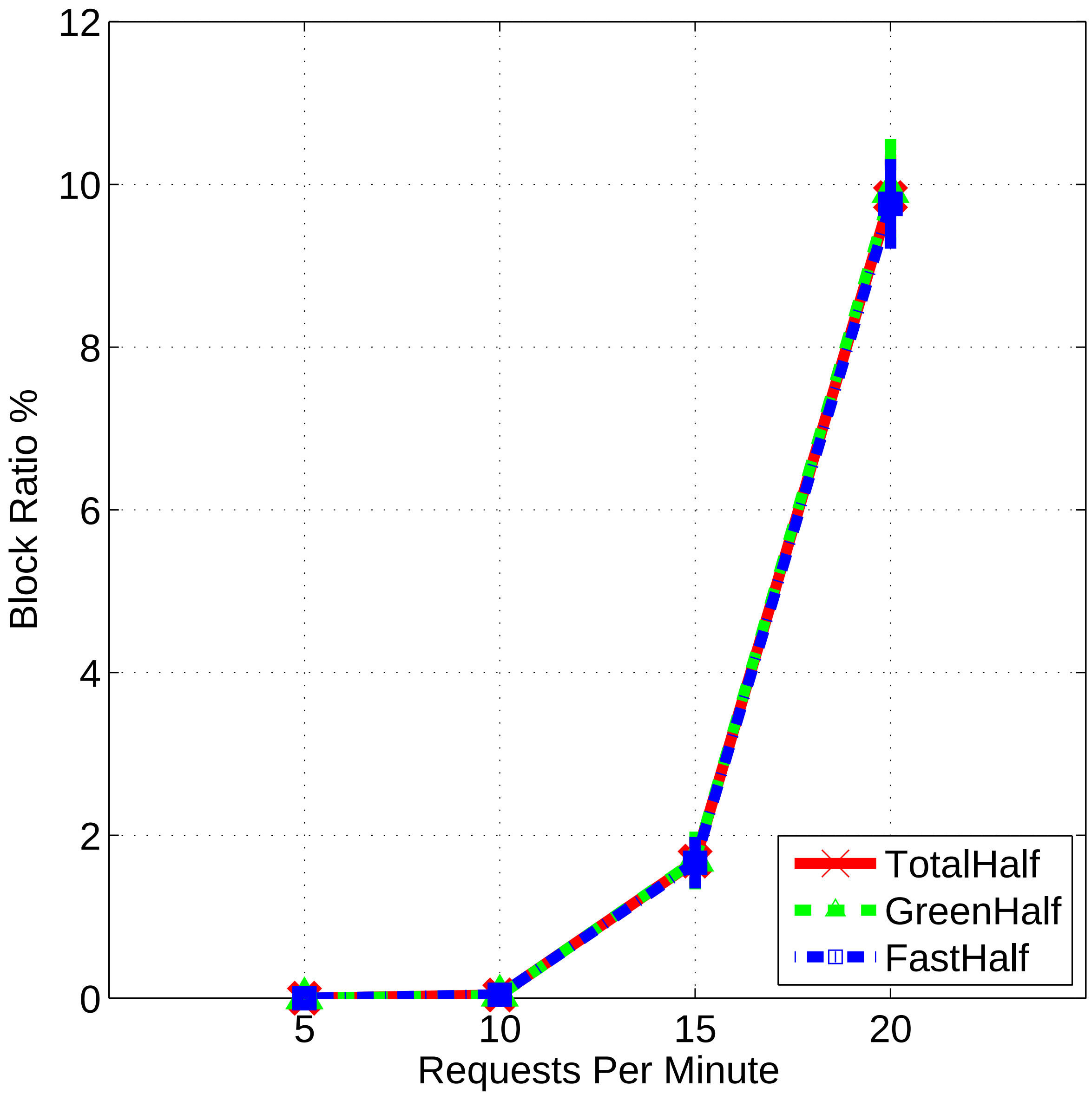


Figure 5.14: HALF case detail: Block Rate

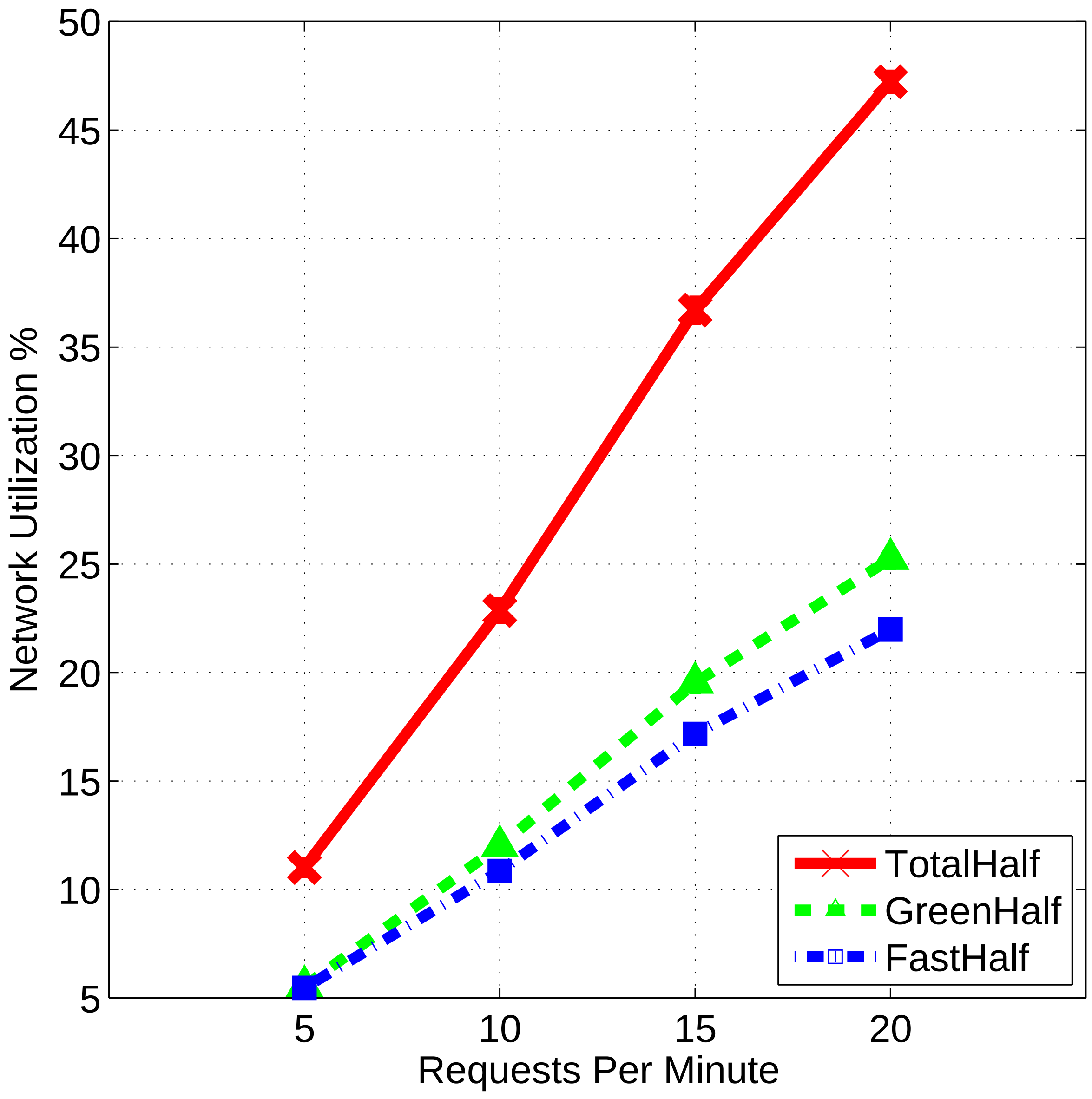


Figure 5.15: HALF case detail: Utilization

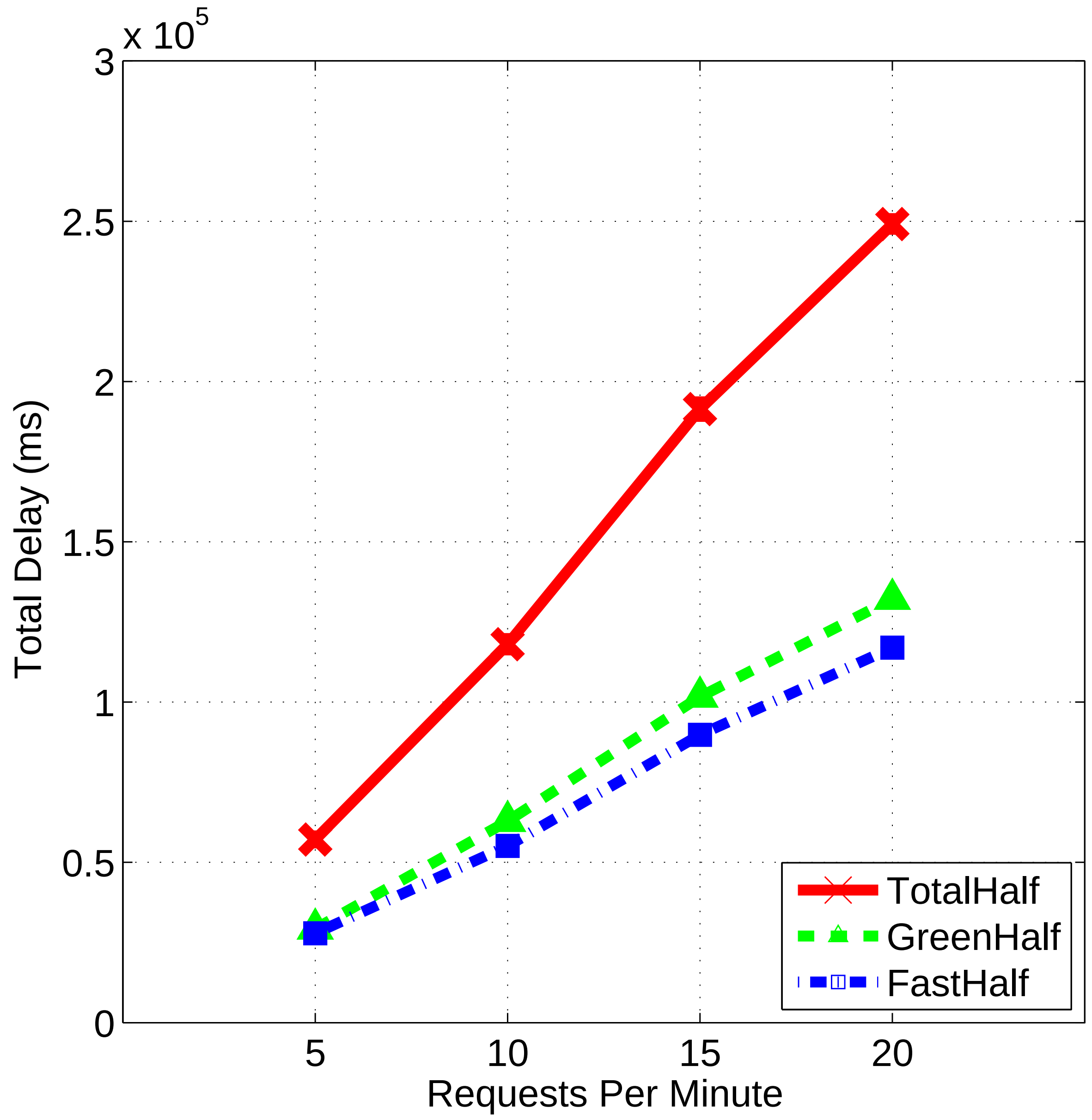


Figure 5.16: HALF case detail: Total Delay

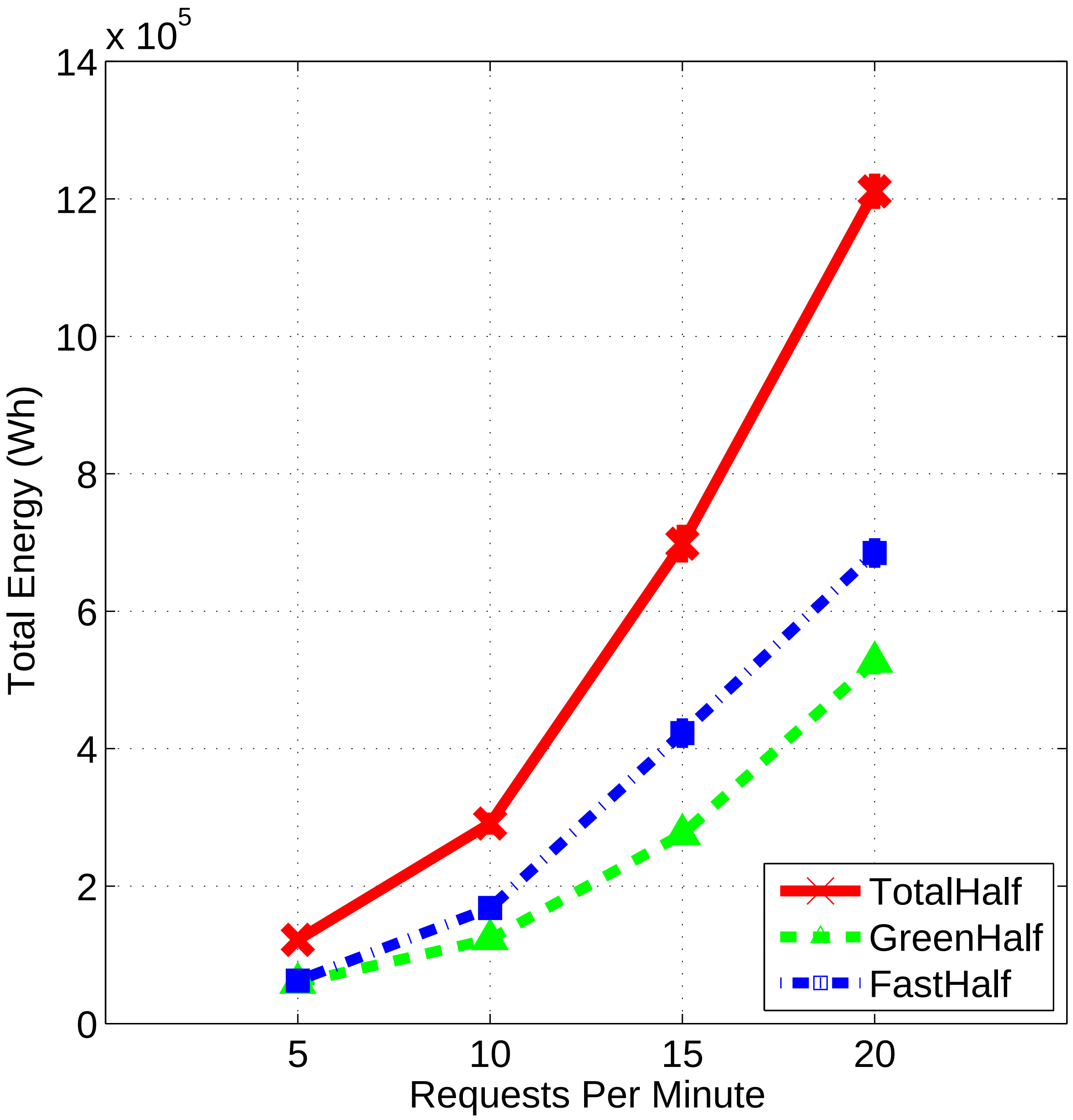


Figure 5.17: HALF case detail: Total Energy

choice-based cooperative optimization. Experimenting with various different parameters can provide more insight into the interaction of various parameters, and the dynamics bewtween customer and provider. For example, Fig. 5.18 shows the Weighted energy metric for a repetition of the simulation runs, but with the optical energy costs artificially perturbed (Wavelength TX and RX Energy Cost increased randomly by a factor of 100, and ADD/DROP ROADM Energy Cost reduced by a factor of 4) . Comparison with Fig. 5.6 shows how much this tends to exaggerate the difference between the various policies. However, their relative nature remains the same. Another apparently puzzling aspect is the reduction of average energy with load for FAST; however, with the exaggerated costs this is expected, since at low load almost each FAST customer will prompt the creation of a new lightpath, imposing heavy energy penalty, whereas at higher loads many arriving FAST customers will find the network well occupied, and be forced to re-use existing lightpaths, thus reducing their energy consumption and forcing down the average.

5.4 Evaluation

Optical backbones provide the bandwidth necessary for all planetary communications, but are often not perceived by the end consumer. As such, the economy, control signaling, and provisioning timescales, have all remained isolated and disconnected between backbone networks providing bulk bandwidth and commodity networks providing service to consumers. This has created barriers for the emergence of innovative and timely, consumer-responsive service offerings; in particular, it has kept the potential energy related benefits of optical networks from being realized. The concept of green grooming can demonstrably help in improving energy efficiency of such a network, but can only be executed by the provider, who then runs the risk of customers dissatisfaction at possible performance degradations.

In this chapter, we have examined a novel mechanism utilizing dynamic choice to form a system that allows providers and consumers to cooperatively enable efficient use of available network resources to mutual benefit. Providers track network resource availability, and pre-compute or compute on demand multiple alternatives to satisfy any given customer demand. Each alternative optimizes different optimization goals such as energy, delay, or other metrics (or a specific balance of metrics), and the price advertised

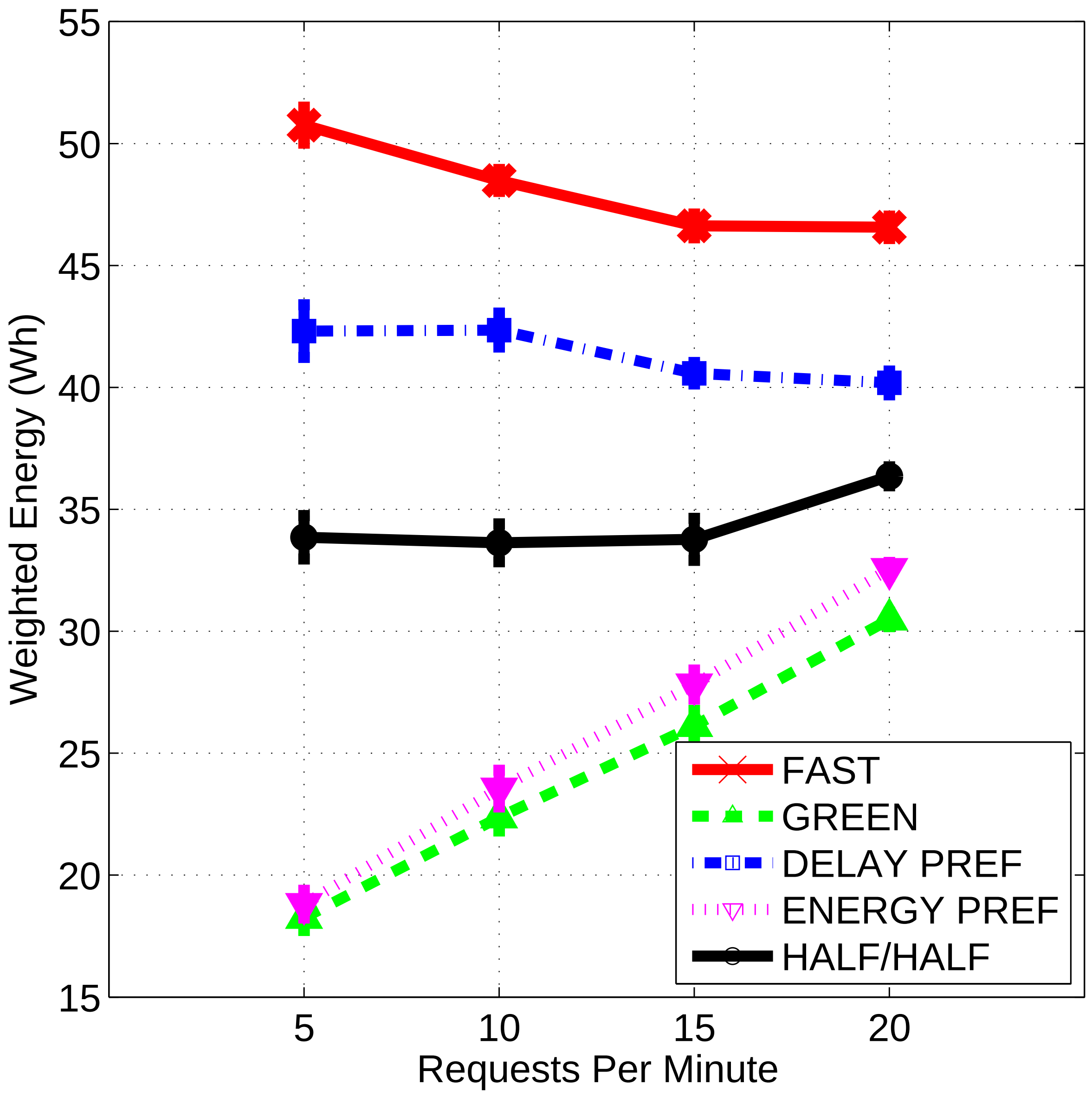


Figure 5.18: Weighted Energy: Perturbed optical energy cost

to the customer indicates the whole cost of operation (including energy) to the provider. We have undertaken a simple abstraction specifying such a system, and our results show that such an approach can indeed improve the performance of the network while being responsive to the different needs of different customers, validating the benefits of using choice-based network services.

Chapter 6

Some Deployment Considerations

In Chapters 3, 4 and 5, we have discussed the Choice-Based Verification Service Architecture along with its performance and benefits. In this chapter, we discuss an incremental and scalable deployment model to increase the appeal of deploying choice-based network services into the current Internet.

The Internet has been tremendously successful in the way it connects people, businesses and access to information in general. Today, users expect reliable, fast and secure access to the information they are looking for. The increasing traffic volume flowing through the Internet combined with the vast variety of user demands makes it attractive to provide on-demand high quality network services rather then the traditional long term best-effort services. On the other hand, the the lack of economical profits in carrying another provider's traffic limits such services across multiple network providers. While we discussed the design and benefits of having a choice-based network architecture in preceding chapters where the user makes agreement with each provider, we believe an incremental and scalable deployment model is essential to increase the appeal of such choice-based network services in real world scenarios. We acknowledge that the content distribution networks today provide a solution by transferring content close to the customers in that geographical area and reducing the number of providers. However, this approach still does not address the issues with real-time communication such as video conferencing or VoIP, where user traffic flows inevitably have to go through very long and uncontrollable distances in some cases. Therefore, the design we discuss in this chapter allows both the network providers to enable such high quality services dynamically and also verification providers to scalably and incrementally verify the user flows. We first describe the "network service gateways" concept, followed by the measurement and verification design. Finally, in Section 6.3, we provide a discussion on the deployment model.

6.1 Network Service Gateways

Today, the commonly agreed communication protocol is IP (Internet Protocol) and any network node using IP can communicate with any other node in the Internet. This has been highly successful in connecting networks in a simple, scalable and stateless way, but it has limited traffic engineering capabilities. Considering billions of users with increased high bandwidth Internet connections, prioritizing individual user flows at the core of the network among millions of flows is (at least economically) not possible. In order to overcome this problem, solutions such as MPLS have been widely used to create tunnels within a provider. While MPLS allows some control over the quality of service, on demand inter-provider MPLS communication has still not been widely deployed due to the questionable economical benefits gained from such interactions for an average user traffic.

Figure 6.1 shows an example scenario of our "network service gateways (NSG)" design located on (or close to) ISPs backbone that create network service tunnels for users who want to use on-demand network services. These tunnel can be established either on-demand or by a long-term manual configuration of each provider along the path. The packets going through tunnel can be defined by any compatible tag, for example an MPLS tag or simply the source and destination IP pair of corresponding NSGs. A single such tunnel can carry all the individual user flows that are interested in using that tunnel and therefore the intermediate providers can easily and scalably identify these flows, treat them with QoS needs and get extra revenue by carrying them. Additionally, in Figure 6.1 the pentagons located between network service providers illustrate the measurement points providing the measurement service, discussed in Chapter 3 and it is more scalable for those providers to filter and measure these packets.

Using the ChoiceNet system (discussed in Chapter 3), the procedure can be described in 5 steps as follows;

Discovery: The customer queries the marketplace to find a tunnel matching its traffic demands. The marketplace returns results as a set of tunnels with corresponding

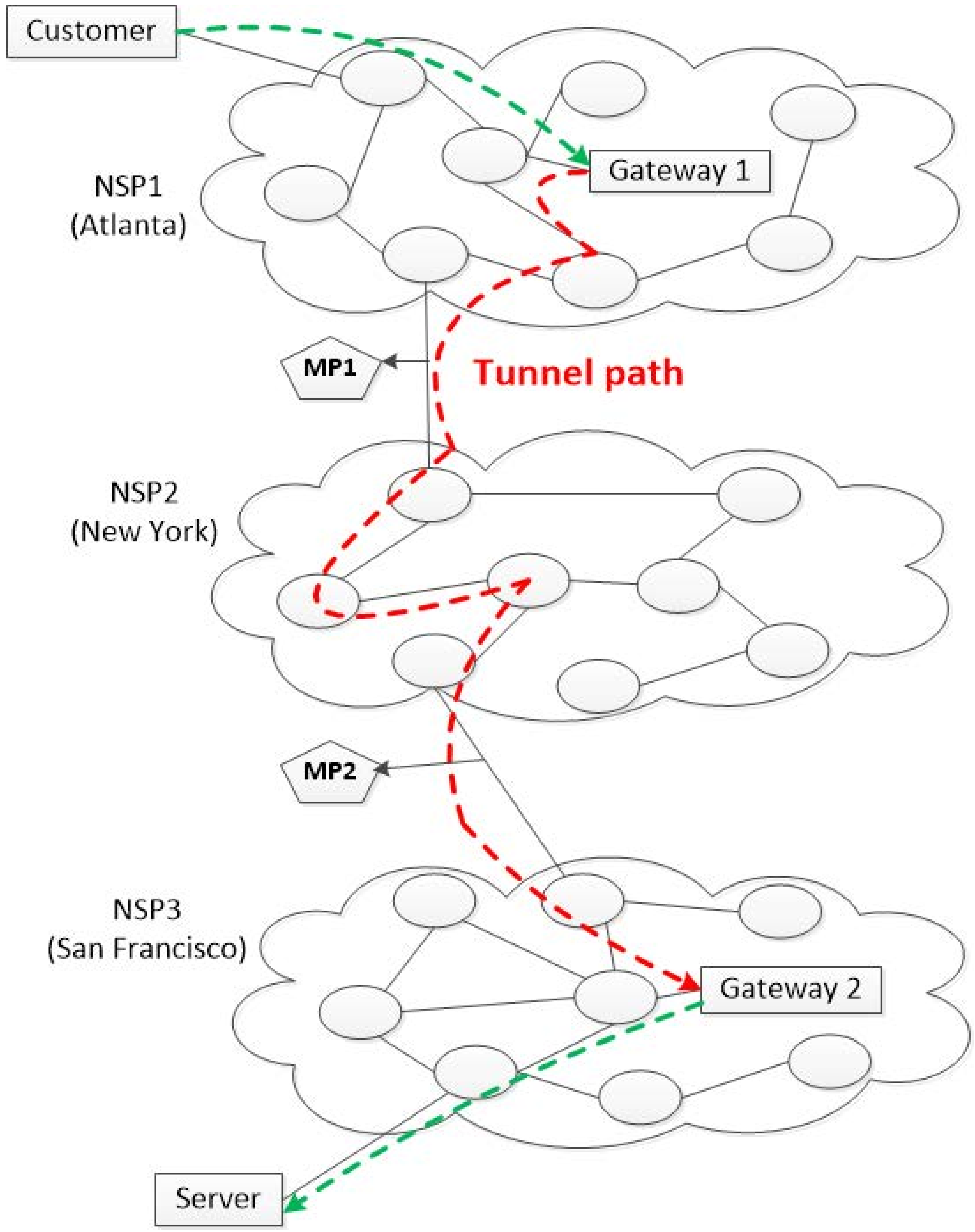


Figure 6.1: An example of Network Service Gateways and Measurement Points

NSG pairs.

Contracts: The customer then connects the source and destination NSGs economy-plane servers (Gateway1 and Gateway2, for NSG1 and NSG2 respectively) of the tunnel and requests for access to a tunnel service with parameters along with the payment. NSG economy-plane servers process and then accept the agreement.

Transmission to source NSG: The customer now starts sending packets encapsulated with destination IP of NSG1's use-plane server. At NSG1's use-plane server, the corresponding machine decapsulates the header, looks for the action rule and encapsulates the packet with an MPLS (or corresponding tag) and the packet goes through the tunnel.

Reception at destination NSG: At the other side of the tunnel, NSG2's corresponding machine decapsulates the packet and looks for the action, the action is to just send the packet to the server. From now on it goes to the server like a normal packet.

Server reply: The server receives the packet and sends a reply. If the contract is only one-way (from customer to server), then server simply sends packets through the best-effort Internet to the customer. An example can be a video streaming scenario, where the video server's traffic must be sent with high quality of service but user's acknowledgements don't need to be sent with such quality. If the contract is in both ways, the server encapsulates and send the packets to NSG2, where they are looked up and sent to NSG1. NSG1 then does the decapsulation and delivers packets to the customer in the same way.

A more detailed view of the gateway is shown in Figure 6.2. The economy plane interaction first takes place to make sure the NSG checks the tunnel state (is the tunnel overloaded? or doing ok?) agrees to send traffic to the NSG for the given quality of service and for that price. NSG then updates its processing machine (NSG use-plane server) to install the new rule and gives the customer the IP address of the processing machine. Now the customer sends packets and processing machine does a table look up to find the action, i.e. the destination NSG and tag for it.

The intermediate providers may or may not know about changing traffic and/or agreements. It might be the case that intermediate providers simply do not have a system to

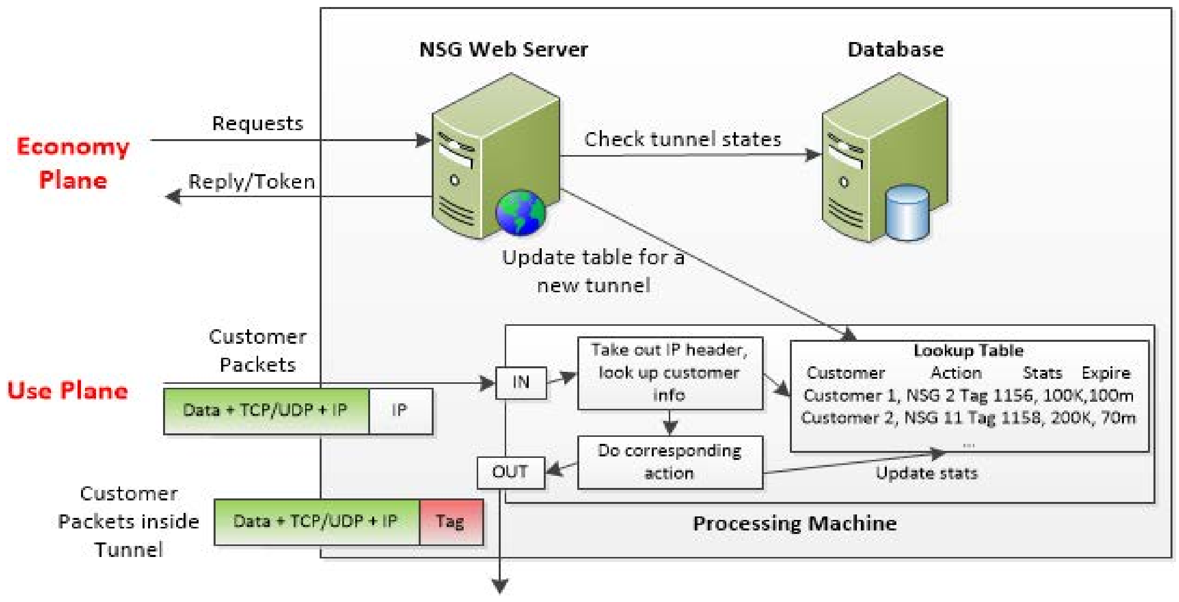


Figure 6.2: Gateway

get informed about changing conditions, however, they can still forward the tunnel traffic tag with high QoS, because they know they will get profit for that. We discuss more about the benefit of this in Section 6.3.

While we advocate for using the gateway to improve the quality of service in terms of low delay, low jitter, high bandwidth, low packet loss etc., other services can also be provided to the users, such as compression, encryption, error-correction, multi-path giving more return value for each NSG deployment.

6.2 Measurement Points and Verification

As discussed in Chapter 3, if the users can make choices about the network services and pay extra for high quality services, they would like to make sure it is worth the investment. Because the customer packets can travel through several providers, it is important to know which provider(s) is(are) causing the problems, so that users can switch to different providers. As shown in Figure 6.1, deploying measurement points (MP) to IXPs can enable us to quickly identify each provider's performance to apportion the overall performance and we have given the design and performance details of a verification service in Chapter 3 and in Chapter 4. However, we have not addressed a scalable and incremental model to filter and process only the relevant customer packets from large amounts of packets.

Today, large IXPs typically carry traffic on the order of Tbps (tera bytes) and considering the annual traffic increase, looking up each and every packet is not economically feasible, at least at the beginning of deployment. Moreover, we are only interested in the performance of the packets going through our tunnels and they can be defined simply by an identifier such as MPLS tag or source/destination IP pair. At the core switches, all we need to do is to add a rule to get a copy of the packets belonging to our tag definition, and we can simply deal with only a small amount of the whole IXP traffic. Once we get access to the tunnel packets, we can process and measure each flow individually via methods described in Chapter 4.

We acknowledge that even after using these tunnels, it is still possible to have quality of service issues between the customer and the source NSG as well as destination NSG to server. However, three things to consider are;

1. We assume the customer and source NSG are geographically close (same city, state, country) so the variation at the delay, jitter and bandwidth values does not change significantly. Typically at the access network with today's broadband technologies (xDSL, cable, fiber), the download or upload bandwidth is not limited by the physical layer but by the provider's rate limiting policy. The provider can instead increase that download and upload limit for packets going to or coming from the NSG's machine IP address, to make sure that is not the limit. Similarly the destination NSG server and destination NSG are geographically closely located.

2. With software defined networking, some providers might actually provide on demand VLAN or MPLS path setup from customer's access network to the NSG, which solves the quality of service problems. Similarly NSG servers can adopt a long term setup with the network service provider to make sure their network link is not the performance bottleneck.

3. Even if such a performance problem occurs in these cases, the measurement point after source NSG (or before destination NSG) will detect this change and report that problem, so we know exactly which provider caused the problem.

6.3 Deployment Model and Further Discussion

Any new technology can only be practical if it is economically feasible to invest at it and this problem only gets worse if a new technology is to be deployed at the core of the Internet because any solution has to be compatible with existing technologies to minimize replacing existing equipment and applications. Also, incremental deployment is essential to reduce the risk of using the technology and lower the barrier to enter this business. Therefore, our deployment model is three-fold;

NSG Deployment: For the network service gateway, the tunnels between different parts of world (especially developing countries) and USA (trans-Pacific or trans-Atlantic) will attract customers seeking high quality services, where there are significant latency and bandwidth limitations. The processing equipment (NSG use-plane servers) can be gradually extended based on the number of customers so there is not a significant upfront cost. Also, the intermediate providers will earn considerable profit based on the number of bytes and packets they carry so there is not a significant initial traffic and related cost for the intermediate providers. Furthermore, the tunnels can typically be setup as long-lasting paths, whereas the individual customer request can be on-demand that is added to the tunnels, which means the intermediate providers do not need to have on-demand path setup capability.

MP Deployment: The measurement points can initially be placed at the busiest IXPs such as DE-CIX, LINX, NY-IIX, AMS-IX, JP-NAP etc, because these locations connect hundreds of network or content providers so a single measurement point deployment on such an IXP allows access to all such providers, reducing the traffic access per provider. We also note that the more tunnels created the more rules need to be searched and verified, increasing the complexity of the system. However, because the tunnels and the bandwidth inside tunnels will gradually increase, the measurement service provider can have enough revenue to do more investment in hardware and software to meet that demand.

Stitching Tunnels: At a later phase of the deployment, the initial tunnels can be stitched together. For example, if there is a tunnel A from New York to London, and there is a tunnel B from San Francisco to Tokyo. These tunnels can be stitched by a different tunnel from New York to San Francisco, making a tunnel from London to Tokyo (through NSGs at London, New York, San Francisco and Tokyo), similar to air flights through airports today. Therefore they can be incrementally deployed on top of the current Internet, at the points where it will be most beneficial.

We acknowledge that there are security issues that must be addressed before the system can be used in practice. For example the traffic between the source user and source NSG as well as destination NSG and the destination user may have a security protocol to provide confidentiality and integrity of the messages. For instance, there may be attackers that send packets to a NSG server that expects packets from a valid user or the user himself/herself might generate more packets than agreed with NSG. Furthermore, NSGs themselves may try to trick the measurement service providers by sending bogus packets pretending to provide high throughput. We believe these issues require comprehensive research on security and therefore creating such novel security algorithms are beyond the scope of this thesis. Also, from the privacy point of view, network service gateways and measurement points do not add any additional privacy concern than what is existing today. It is up to the end points to provide strong encryption of application data.

We envision this design allows scalable, incremental, economical and yet economical way to provide high quality service between geographically distant users, especially demanding video conferencing and VoIP applications, which can increase the appeal to deploy choice-based services.

Chapter 7

Conclusions

The future of the Internet is constantly putting new types of demands with increasing volume, and the network architecture research is finding new ways to address these challenges in the most feasible way. In this thesis, we proposed a solution to a missing piece of the Internet, the measurement and verification of the individual network service provider performances and if there are problems in the network, a system that finds the provider(s) that cause(s) the problems.

We have first went through the literature to review the efforts that attempted to solve related problems and then described the fundamentals of our design and interactions between the components. Next, we have shown the arc of the studies that evolved into our verification architecture along with the experiments we have conducted. We have then shown the applicability of our design by simulation and proof of concept prototype on a well-known actual network testbed, GENI.

Finally, we have provided simulation results about the benefits of using choice-based networks from the energy-efficiency view in optical networks, followed by a discussion on incremental and scalable deployment model of our architecture.

There is certainly more work to realize a complete real verification service that actually runs as an integral part of the current Internet. We envision there are at least three important points to address before this system can actually be used on the current Internet;

Security: Cases must be handled where providers or other users may create bogus packets to fool measurement providers, users may not behave appropriately or the measurement service providers may give false or significantly inaccurate results (review based systems). This will make the system more robust in practical cases.

Approximation: With less number of service providers, can we gather more information to conclude the results? This will greatly help with the incremental deployments. Note that this is more to do with the analysis of measurement data.

The actual deployment: starting measurement points at campus network or measurement point for GENI itself, later followed by parts of real IXPs.

In this interesting era of networks, where software defined networks are starting to become widely used, enabling choices at the core of the Internet will yet create new opportunities to come and change the way we understand, use and experience network functions and providers.

REFERENCES

[1] G. Aceto, A. Botta, A. Pescape, and M. D"Arienzo. UANM: a platform for experimenting with available bandwidth estimation tools. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 174–179, June.

[2] Bernhard Ager, Nikolaos Chatzis, Anja Feldmann, Nadi Sarrar, Steve Uhlig, and Walter Willinger. Anatomy of a large european IXP. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '12, pages 163–174, New York, NY, USA, 2012. ACM.

[3] Mohammad Z. Ahmad and Ratan Guha. Understanding the impact of internet exchange points on internet topology and routing performance. In *Proceedings of the ACM CoNEXT Student Workshop*, CoNEXT '10 Student Workshop, pages 18:1–18:2, New York, NY, USA, 2010. ACM.

[4] I. Aktas, J. Otten, F. Schmidt, and K. Wehrle. Towards a flexible and versatile cross-layer-coordination architecture. In *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, pages 1–5, March.

[5] A. Alimian, B. Nordman, and D. Kharitonov. Network and Telecom Equipment Energy and Performance Assessment - Test Procedures and Measurement Methodology, ECR Initiative, 2008. `http://www.ecrinitiative.org/pdfs/ECR_1_0_2.pdf`, October 2008.

[6] Mark Allman and Vern Paxson. A reactive measurement framework. In *Proceedings of the 9th international conference on Passive and active network measurement*, PAM'08, pages 92–101, Berlin, Heidelberg, 2008. Springer-Verlag.

[7] L. Angrisani and C. Narduzzi. Measurements for networking: An overview. In *Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE*, pages 1328–1333, May.

[8] A.C. Babaoglu and R. Dutta. A verification service architecture for the future internet. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, pages 1–9, July 2013.

[9] Ahmet Can Babaoglu. Architectural considerations in integrating measurement plane interaction in the SILO architecture. `http://www4.ncsu.edu/~acbabaog/csc890_report_acbabaog.pdf`, 2010.

[10] Ahmet Can Babaoglu. Networking Frontiers Presentation. `http://www4.ncsu.edu/~acbabaog/csc401_lecture.pdf`, 2011.

[11] Ahmet Can Babaoglu and Rudra Dutta. Performance impact of architectural decisions: integrating measurement in SILO. In *Proceedings of the 6th International Conference on Future Internet Technologies*, CFI '11, pages 72–78, New York, NY, USA, 2011. ACM.

[12] F. Baker. *RFC 1812 Requirements for IP Version 4 Routers*, 1995.

[13] I. Baldine, M. Vellala, Anjing Wang, G. Rouskas, R. Dutta, and D. Stevenson. A unified software architecture to enable cross-layer design in the future internet. In *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, pages 26–32, Aug.

[14] J. Baliga, K. Hinton, and RodneyS. Tucker. Energy consumption of the internet. In *Optical Internet, 2007 and the 2007 32nd Australian Conference on Optical Fibre*

Technology. COIN-ACOFT 2007. Joint International Conference on, pages 1–3, 2007.

[15] Mark Berman, Jeffrey S. Chase, Lawrence Landweber, Akihiro Nakao, Max Ott, Dipankar Raychaudhuri, Robert Ricci, and Ivan Seskar. Geni: A federated testbed for innovative network experiments. *Computer Networks*, 2014.

[16] E. Blanton, S. Chatterjee, S. Gangam, S. Kala, D. Sharma, S. Fahmy, and P. Sharma. Design and evaluation of the s3 monitor network measurement service on geni. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pages 1–10, Jan.

[17] Marjory S. Blumenthal and David D. Clark. Rethinking the design of the internet: the end-to-end arguments vs. the brave new world. *ACM Trans. Internet Technol.*, 1(1):70–109, August 2001.

[18] Jean Chrysotome Bolot. End-to-end packet delay and loss behavior in the internet. *SIGCOMM Comput. Commun. Rev.*, 23(4):289–298, October 1993.

[19] Nicola Bonelli, Andrea Di Pietro, Stefano Giordano, and Gregorio Procissi. On multi—gigabit packet capturing with multi—core commodity hardware. In *Proceedings of the 13th international conference on Passive and Active Measurement*, PAM'12, pages 64–73, Berlin, Heidelberg, 2012. Springer-Verlag.

[20] Tierney Boote, Boyd Brown, Grigoriev Metzger, Swany Zekauskas, and Li Zurawski. Instantiating a Global Network Measurement Framework. *LBNL Technical Report LBNL-1452E*, 2009.

[21] Robert Braden, Ted Faber, and Mark Handley. From protocol stack to protocol heap: role-based architecture. *SIGCOMM Comput. Commun. Rev.*, 33(1):17–22, January 2003.

[22] Justin Cappos, Ivan Beschastnikh, Arvind Krishnamurthy, and Tom Anderson. Seattle: a platform for educational cloud computing. In *Proceedings of the 40th ACM technical symposium on Computer science education*, SIGCSE '09, pages 111–115, New York, NY, USA, 2009. ACM.

[23] Robert L. Carter and Mark E. Crovella. Measuring bottleneck link speed in packet-switched networks. *Perform. Eval.*, 27-28:297–318, October 1996.

[24] J. Case, M. Fedor, M. Schoffstall, and J. Davin. *RFC 1157: A Simple Network Management Protocol*, 1990.

[25] CentMesh Website. `http://centmesh.csc.ncsu.edu/`.

[26] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power awareness in network design and routing. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages –, 2008.

[27] ChoiceNet: Network Innovation through Choice, Website. `http://code.renci.org/gf/project/choicenet/`.

[28] CISCO. NetFLOW Export Datagram Format. `http://www.cisco.com/en/US/docs/net_mgmt/netflow_collection_engine/3.6/user/guide/format.html`.

[29] CISCO. NetFLOW Case Study White Paper. `http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_case_study0900aecd80311fc2.pdf`, 2003.

[30] CISCO. Cisco NetFLOW Performance Analysis. `http://www.cisco.com/en/US/technologies/tk543/tk812/technologies_white_paper0900aecd802a0eb9.html`, 2007.

[31] CISCO. Introduction to Cisco IOS NetFlow white paper. `http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_white_paper0900aecd80406232.html`, 2012.

[32] B. Claise. *RFC 3954: Cisco Systems NetFlow Services Export Version 9*, October 2004.

[33] B. Claise. *RFC 5101, Specification of the IP Flow Information Export (IPFIX) Protocol*, 2008.

[34] D. Clark. The design philosophy of the DARPA internet protocols. *SIGCOMM Comput. Commun. Rev.*, 18(4):106–114, August 1988.

[35] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. *SIGCOMM Comput. Commun. Rev.*, 20(4):200–208, August 1990.

[36] David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. Tussle in cyberspace: defining tomorrow's internet. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '02, pages 347–356, New York, NY, USA, 2002. ACM.

[37] J. Cleary, S. Donnelly, I. Graham, A. Mcgregor, and M. Pearson. Design principles for accurate passive measurement. In *The First Passive and Active Measurement Workshop*, pages 1–7, Hamilton, New Zealand, April 2000.

[38] Daniele Croce, Marco Mellia, and Emilio Leonardi. The quest for bandwidth estimation techniques for large-scale distributed systems. *SIGMETRICS Perform. Eval. Rev.*, 37(3):20–25, January 2010.

[39] Endace DAG Cards Website. `http://www.endace.com/endace-dag-high-speed-packet-capture-cards.html`.

[40] C. Demichelis and P. Chimento. IP Packet Delay Variation Metric for IP Performance Metrics (IPPM), November 2002.

[41] Allen B. Downey. Clink: a tool for estimating Internet link characteristics. `http://allendowney.com/research/clink/`.

[42] Allen B. Downey. Using pathchar to estimate Internet link characteristics. *SIGCOMM Comput. Commun. Rev.*, 29(4):241–250, August 1999.

[43] R. Dutta and G.N. Rouskas. Traffic grooming in wdm networks: past and future. *Network, IEEE*, 16(6):46–56, Nov 2002.

[44] Rudra Dutta, Ahmed E. Kamal, and George N. Rouskas, editors. *Traffic Grooming for Optical Networks: Foundations, Techniques and Frontiers.* Springer, 2008 edition, 8 2008.

[45] Rudra Dutta, George N. Rouskas, Ilia Baldine, Arnold Bragg, and Dan Stevenson. The SILO architecture for services integration, control, and optimization for the future internet. In *IEEE ICC*, pages 24–27, 2007.

[46] Aaron Falk. GENI System Overview. Technical report, The GENI Project Office, 2008.

[47] Anja Feldmann. Internet clean-slate design: what and why? *SIGCOMM Comput. Commun. Rev.*, 37(3):59–64, July 2007.

[48] P. Ferrari, A. Flammini, S. Rinaldi, A. Bondavalli, and F. Brancati. Evaluation of timestamping uncertainty in a software-based ieee1588 implementation. In *Instrumentation and Measurement Technology Conference (I2MTC), 2011 IEEE*, pages 1–6, May.

[49] NSF Future Internet Architecture (FIA) Project. `http://www.nets-fia.net/`.

[50] Future Internet Architecture (FIA) Workshop, Nov 2013, San Diego, CA. `http://www.eventbrite.com/e/nov-1415-2013-fia-workshop-ucsd-tickets-8458057277?utm_source=eb_email&utm_medium=email&utm_campaign=email_attendees&utm_term=event`.

[51] National Science Foundation, NSF FIND (Future Internet Design). `http://www.nets-find.net`.

[52] Ross Finlayson. Live555 Streaming Media. `http://www.live555.com/liveMedia`.

[53] EU FIRE (Future Internet Research and Experimentation) Testbed Project. `http://cordis.europa.eu/fp7/ict/fire/home_en.html`.

[54] Darleen Fisher. Us national science foundation and the future internet design. *SIGCOMM Comput. Commun. Rev.*, 37(3):85–87, July 2007.

[55] F. Fitzek and M. Reisslein. Mpeg-4 and h.263 video traces for network performance evaluation. *Network, IEEE*, 15(6):40–54, 2001.

[56] GENI Flack Website. `http://www.protogeni.net/wiki/Flack`.

[57] ORCA Flukes Tool. http://geni-orca.renci.org/trac/wiki/flukes.

[58] The National Laboratory for Applied Network Research/The Distributed Applications Support Team (NLANR/DAST). Iperf. http://iperf.sourceforge.net.

[59] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S.C. Diot. Packet-level traffic measurements from the sprint ip backbone. *Network, IEEE*, 17(6):6–16, Nov.-Dec.

[60] Chuck Fraleigh, Christophe Diot, Bryan Lyles, Sue B. Moon, Philippe Owezarski, Dina Papagiannaki, and Fouad A. Tobagi. Design and deployment of a passive monitoring infrastructure. In *Proceedings of the Thyrrhenian International Workshop on Digital Communications: Evolutionary Trends of the Internet*, IWDC '01, pages 556–575, London, UK, UK, 2001. Springer-Verlag.

[61] Brynjar ge Viken. Passive monitoring of internet traffic at supercomputing '98. In *Proceedings of EUNICE '99*, 1999.

[62] GEMINI: A GENI Measurement and Instrumentation Infrastructure. http://groups.geni.net/geni/wiki/GEMINI.

[63] BBN Technologies, GENI: Global Environment for Network Innovations. http://www.geni.net.

[64] GIMI: Large-scale GENI Instrumentation and Measurement Infrastructure. http://groups.geni.net/geni/wiki/GIMI.

[65] Glassman S. and Manasse M. and Abadi M. and Gauthier P. and Sobalvarro P. The Millicent protocol for inexpensive electronic commerce. *In World Wide Web Journal, Fourth International World Wide Web Conference Proceedings*, 1995.

[66] Ian D. GRAHAM, Stephen F. DONNELLY, Stele MARTIN, Jed MARTENS, and John G. CLEARY. Nonintrusive and accurate measurement of unidirectional delay and delay variation on the internet. In *INET*, 1998.

[67] J. Griffioen, Z. Fei, and H. Nasir. Architectural Design and Specification of the INSTOOLS Measurement System. Technical report, University of Kentucky, 2009.

[68] Andreas Hanemann, Jeff W. Boote, Eric L. Boyd, Jérôme Durand, Loukik Kudarimoti, Roman Lapacz, D. Martin Swany, Szymon Trocha, and Jason Zurawski. PerfSONAR: a service oriented architecture for multi-domain network monitoring. In *Proceedings of the Third international conference on Service-Oriented Computing*, ICSOC'05, pages 241–254, Berlin, Heidelberg, 2005. Springer-Verlag.

[69] Shu Huang and R. Dutta. Dynamic traffic grooming: the changing role of traffic grooming. *Communications Surveys Tutorials, IEEE*, 9(1):32–50, First 2007.

[70] Shu Huang and R. Dutta. Dynamic traffic grooming: the changing role of traffic grooming. *Communications Surveys Tutorials, IEEE*, 9(1):32–50, 2007.

[71] Shu Huang, D. Seshadri, and R. Dutta. Traffic grooming: A changing role in green optical networks. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1–6, 2009.

[72] Shu Huang, D. Seshadri, and R. Dutta. Traffic grooming: A changing role in green optical networks. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1–6, Nov 2009.

[73] Norman C. Hutchinson and Larry L. Peterson. The x-kernel: An architecture for implementing network protocols. *IEEE Trans. Softw. Eng.*, 17(1):64–76, January 1991.

[74] IEEE 1588 Precise Time Protocol: The new standard in time synchronization White Paper. http://www.symmetricom.com/resources/downloads/white-papers/ IEEE-1588-PTP-Solutions/IEEE-1588-Precise-Time- Protocol-The-New-Standard-in-Time-Synchronization/, 2005.

[75] North Carolina State University and Columbia University and University of North Carolina at Chapel Hill and RENCI and University of Houston, IMF (Integrated Measurement Framework) RENCI Website. `http://geni-imf.renci.org`.

[76] North Carolina State University and Columbia University and University of North Carolina at Chapel Hill and RENCI and University of Houston, IMF (Integrated Measurement Framework) GENI Website. `http://groups.geni.net/geni/wiki/IMF`.

[77] InMON. sFlow Website. `http://sflow.org/`.

[78] iRODS (integrated Rule-Oriented Data System) website. `http://www.irods.org`.

[79] Van Jacobson. traceroute. `ftp://ftp.ee.lbl.gov/traceroute.tar.gz`.

[80] Van Jacobson. pathchar – a tool to infer characteristics of Internet paths. Slides available from `ftp://ee.lbl.gov/pathchar/`, April 1997.

[81] Manish Jain and Constantinos Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth. *In Proceedings of Passive and Active Measurements (PAM) Workshop*, 2002.

[82] R. Jain. Internet 3.0: Ten Problems with Current Internet Architecture and Solutions for the Next Generation. In *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pages 1–9, Oct.

[83] Keith W. Ross James F. Kurose. *Computer Networking: A Top-Down Approach, 4/E*. Addison-Wesley, 2008.

[84] Elisa Jasinska. SFlow: I can feel your traffic, White Paper. `http://www.sflow.org/using_sflow/application_notes.php`, 2006.

[85] Nina Jeliazkova, Luchesar Iliev, and Vedrin Jeliazkov. UPerfsonarUI - a Standalone Graphical User Interface for Querying perfSONAR Services. In *Proceedings of the IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing*, JVA '06, pages 77–81, Washington, DC, USA, 2006. IEEE Computer Society.

[86] JAPANESE JGN2PLUS Testbed Project. `http://www.jgn.nict.go.jp/jgn2plus_archive/english/index.html`.

[87] Andreas Johnsson. On the comparison of packet pair and packet train measurements. In *Swedish National Computer Networking Workshop, Arlandastad*, 2003.

[88] Sunil Kalidindi and Matthew J. Zekauskas. Surveyor: An infrastructure for internet performance measurements. In *INET*, 1999.

[89] A. Karouia, R. Langar, Thi-Mai-Trang Nguyen, and G. Pujolle. Soa-based approach for the design of the future internet. In *Communication Networks and Services Research Conference (CNSR), 2010 Eighth Annual*, pages 361–368, May.

[90] Isaac Keslassy, Shang-Tse Chuang, Kyoungsik Yu, David Miller, Mark Horowitz, Olav Solgaard, and Nick McKeown. Scaling internet routers using optics. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 189–200, New York, NY, USA, 2003. ACM.

[91] Ken Keys, David Moore, Ryan Koga, Edouard Lagache, Michael Tesch, and k claffy. The Architecture of CoralReef: An Internet Traffic Monitoring Software Suite. In *In PAM*, 2001.

[92] Charles M. Kozierok. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference.* No Starch Press, 1 edition, October 2005.

[93] K.Zhu and B.Mukherjee. A review of traffic grooming in wdm optical networks: architectures and challenges. *Optical Networks Magazine*, 4(2):55–64, Mar 2003.

[94] K. Lai and M. Baker. Measuring bandwidth. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 235 –245 vol.1, mar 1999.

[95] Kevin Lai and Mary Baker. Measuring link bandwidths using a deterministic model of packet delay. *SIGCOMM Comput. Commun. Rev.*, 30(4):283–294, August 2000.

[96] Kevin Lai and Mary Baker. Measuring link bandwidths using a deterministic model of packet delay. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '00, pages 283–294, New York, NY, USA, 2000. ACM.

[97] Kevin Lai and Mary Baker. Nettimer: a tool for measuring bottleneck link, bandwidth. In *Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems - Volume 3*, USITS'01, pages 11–11, Berkeley, CA, USA, 2001. USENIX Association.

[98] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff. A brief

history of the internet. *SIGCOMM Comput. Commun. Rev.*, 39(5):22–31, October 2009.

[99] Hanemann Liakopoulos and Molina Swany. A Study on Network Performance Metrics and their Composition. *TERENA Networking Conference*, 2006.

[100] M. A. Lombardi. Time and frequency measurements using the global positioning system (gps). In *Proc. Measurement Science Conference, Anaheim, CA, USA, Jan. 2001*, 2001.

[101] Long Long and A.E. Kamal. p2-cycles: p-cycles with parasitic protection links. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5, 2010.

[102] Matthew J. Luckie, Anthony J. McGregor, and Hans-Werner Braun. Towards improving packet probing techniques. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, pages 145–150, New York, NY, USA, 2001. ACM.

[103] LYNX, Text browser for World Wide Web.

[104] MadWifi Driver. `http://madwifi-project.org/`.

[105] Ratul Mahajan, Neil Spring, David Wetherall, and Thomas Anderson. User-level internet path diagnosis. *SIGOPS Oper. Syst. Rev.*, 37(5):106–119, October 2003.

[106] Keith Marzullo and Susan Owicki. Maintaining the time in a distributed system. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, PODC '83, pages 295–305, New York, NY, USA, 1983. ACM.

[107] Steven McCanne and Van Jacobson. The BSD packet filter: a new architecture for user-level packet capture. In *Proceedings of the USENIX Winter 1993 Conference*

Proceedings on USENIX Winter 1993 Conference Proceedings, USENIX'93, pages 2–2, Berkeley, CA, USA, 1993. USENIX Association.

[108] T. McGregor, H.-W. Braun, and J. Brown. The NLAMR network analysis infrastructure. *Communications Magazine, IEEE*, 38(5):122 –128, may 2000.

[109] Tony McGregor and Hans-Werner Braun. Balancing cost and utility in active monitoring: The amp example. In *INET*, 2000.

[110] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[111] Olivier Mehani, Guillaume Jourjon, Thierry Rakotoarivelo, and Max Ott. An instrumentation framework for the critical task of measurement collection in the future internet. Technical Report 6065, Nicta, Eveleigh, Sydney, NSW, Australia, October 2012.

[112] David L. Mills. *RFC 1305, Network Time Protocol(Version 3)*, 1992.

[113] David L. Mills. *RFC 5905, Network Time Protocol(Version 4)*, 2010.

[114] Mobility First Project Website. `http://mobilityfirst.winlab.rutgers.edu/`.

[115] E. Modiano. Traffic grooming in wdm networks. *Communications Magazine, IEEE*, 39(7):124–129, Jul 2001.

[116] H. Mussman. ORCA GENI Control Framework Overview. Technical report, The GENI Project Office, 2009.

[117] Ahmet Can Babaoglu, Green Grooming Code Implementation, http://www4.ncsu.edu/~acbabaog/choiceBasedGroomingCode.zip.

[118] B. Nordman, Networks, Energy, and Energy Efficiency, http://groups.geni.net/geni/attachment/wiki/GEC1/Bruce_Nordman_Networks_Energy_and_Energy_Efficiency.pdf.

[119] Jad Naous, Michael Walfish, Antonio Nicolosi, David Mazires, Michael Miller, and Arun Seehra. Verifying and enforcing network paths with icing. In *in Proceedings of ACM CoNEXT*, 2011.

[120] Named Data Networking (NDN) Project Website. http://www.named-data.net/.

[121] NEBULA Cloud-based Network Architecture Project Website. http://nebula.cis.upenn.edu/.

[122] NEBULA Cloud-based Network Architecture Presentation. http://www.nets-fia.net/Meetings/Nov10/Kickoff-public-Nebula.pdf.

[123] D.T. Neilson. Photonics for switching and routing. *Selected Topics in Quantum Electronics, IEEE Journal of*, 12(4):669–678, 2006.

[124] Netflix Super HD Website.

[125] NS3 Simulator. http://www.nsnam.org/.

[126] OMF (cOntrol and Management Framework. http://omf.mytestbed.net/.

[127] OpenFlow Specification Version 1.0. http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf.

[128] OpenFlow-Enabled IBM G8264 Switch Specifications. `http://www.openflow.org/wp/ibm-switch/`.

[129] OpenVswitch Website. `http://openvswitch.org/`.

[130] OpenFlow Website. `www.openflow.org`.

[131] The Renaissance Computing Institute (RENCI), BEN (Breakable Experimental Network) Project Website. `https://ben.renci.org/index.php?Itemid=84`.

[132] The Renaissance Computing Institute (RENCI), ORCA (Open Resource Control Architecture) Project Website. `http://geni-orca.renci.org`.

[133] V. Paxson, A.K. Adams, and M. Mathis. Experiences with NIMI. In *Applications and the Internet (SAINT) Workshops, 2002. Proceedings. 2002 Symposium on*, pages 108 –118, 2002.

[134] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. RFC 2330: Framework for IP performance metrics, May 1998.

[135] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis. An architecture for large scale Internet measurement. *Comm. Mag.*, 36(8):48–54, August 1998.

[136] T.S. Perry. Fueling the internet [computer power consumption]. *Spectrum, IEEE*, 38(1):80–82, 2001.

[137] An Extensible Schema for Network Measurement and Performance Data, Dr. Martin Swany, May 2007. `http://nmwg.internet2.edu/nm-schema-base.html`.

[138] J. Postel. *RFC 792 Internet Control Message Protocol*, 1981.

[139] Jon Postel. *RFC 791 Internet Protocol*, 1981.

[140] R. S. Prasad, M. Murray, C. Dovrolis, K. Claffy, Ravi Prasad, and Constantinos Dovrolis Georgia. Bandwidth estimation: Metrics, measurement techniques, and tools. *IEEE Network*, 17:27–35, 2003.

[141] J. Quittek, S. Bryant, B. Claise, P. Aitken, and J. Meyer. *RFC 5102, Information Model for IP Flow Information Export (IPFIX)*, 2008.

[142] J. Quittek, T. Zseby, B. Claise, and S. Zander. *RFC 3917, Requirements for IP Flow Information Export (IPFIX)*, 2004.

[143] M. Rabinovich, S. Triukose, Zhihua Wen, and Limin Wang. Dipzoom: The internet measurements marketplace. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–6, April.

[144] Glauco Caurin Rafael V. Aroca1. A Real Time Operating Systems (RTOS) Comparison. In *Workshop de Sistemas Operacionais (Operating Systems) (WSO'2009)*, 2009.

[145] Vijay T Raisinghani and Sridhar Iyer. Cross-layer design optimizations in wireless protocol stacks. *Computer Communications*, 27(8):720 – 724, 2004. ¡ce:title¿Advances in Future Mobile/Wireless Networks and Services¡/ce:title¿.

[146] Thierry Rakotoarivelo, Maximilian Ott, Guillaume Jourjon, and Ivan Seskar. Omf: a control and management framework for networking testbeds. *SIGOPS Oper. Syst. Rev.*, 43(4):54–59, January 2010.

[147] Jennifer Rexford and Constantine Dovrolis. Future internet architecture: clean-slate versus evolutionary research. *Commun. ACM*, 53(9):36–40, September 2010.

[148] Vinay J. Ribeiro, Rudolf H. Riedi, Richard G. Baraniuk, Jiri Navratil, and Les Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. *Proceedings of the 4th Passive and Active Measurement Workshop (PAM 2003)*, 2003.

[149] Ian Rose and Matt Welsh. Mapping the urban wireless landscape with Argos. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 323–336, New York, NY, USA, 2010. ACM.

[150] G. Rouskas, R. Dutta, D. Stevenson, I. Baldine, A. Wang, and M. Vellala. SILO Project. `http://www.net-silos.net`.

[151] Rouskas, George N. and Baldine, Ilia and Calvert, Kenneth L. and Dutta, Rudra and Griffioen, Jim and Nagurney, Anna and Wolf, Tilman. ChoiceNet: Network Innovation Through Choice. *Optical Network Design and Modeling (ONDM), 2013 17th International Conference on*, 2013.

[152] RTAI (Real Time Application Interface) Website. `http://www.rtai.org/`.

[153] G. Sadasivan, N. Brownlee, B. Claise, and J. Quittek. *RFC 5470, Architecture for IP Flow Information Export (IPFIX)*, 2009.

[154] Peter Saint-Andre, Kevin Smith, and Remko TronCon. *XMPP: The Definitive Guide: Building Real-Time Applications with Jabber Technologies.* O'Reilly Media, 1 edition, 5 2009.

[155] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, November 1984.

[156] L. Sampaio, I. Koga, R. Costa, H. Monteiro, J.A.S. Monteiro, F. Vetter, G. Fernandes, and M. Vetter. Implementing and deploying network monitoring service oriented architectures: Brazilian national education and research network measurement experiments. In *Network Operations and Management Symposium, 2007. LANOMS 2007. Latin American*, pages 28–37, Sept.

[157] Dheeraj Sanghi, Olafur Gudmundsson, and Ashok K. Agrawala. Study of network dynamics. *Computer Networks and ISDN Systems*, 26(3):371 – 378, 1993.

[158] Stefan Savage. Sting: a TCP-based network measurement tool. In *Proceedings of the 2nd conference on USENIX Symposium on Internet Technologies and Systems - Volume 2*, USITS'99, pages 7–7, Berkeley, CA, USA, 1999. USENIX Association.

[159] Fabian Schneider, Jörg Wallerich, and Anja Feldmann. Packet capture in 10-gigabit ethernet environments using contemporary commodity hardware. In *Proceedings of the 8th international conference on Passive and active network measurement*, PAM'07, pages 207–217, Berlin, Heidelberg, 2007. Springer-Verlag.

[160] Seattle Educational Cloud Computing Website. `https://seattle.cs.washington.edu/html/`.

[161] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Flowvisor: A network virtualization layer. Technical report, Deutsche Telekom Inc. R&D Lab and Stanford University and Nicira Networks, 2009.

[162] K. Shimizu, K. Sebayashi, S. Kuwabara, and M. Maruyama. 10-gbps ip network measurement system based on application-generated packets using hardware assistance and off-the-shelf pc. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6, June.

[163] Alok Shriram, Margaret Murray, Young Hyun, Nevil Brownlee, Andre Broido, Marina Fomenkov, and kc claffy. Comparison of public end-to-end bandwidth estimation tools on high-speed links. In *Proceedings of the 6th international conference on Passive and Active Network Measurement*, PAM'05, pages 306–320, Berlin, Heidelberg, 2005. Springer-Verlag.

[164] Murray R Spiegel, John J. Schiller, and R. Alu Srinivasan. *Schaum's Outline: Probability and Statistics, Second Edition.* McGraw-Hill, 2nd edition, 3 2000.

[165] Neil Spring, David Wetherall, and Tom Anderson. Scriptroute: a public internet measurement facility. In *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems - Volume 4*, USITS'03, pages 17–17, Berkeley, CA, USA, 2003. USENIX Association.

[166] V. Srivastava and M. Motani. Cross-layer design: a survey and the road ahead. *Communications Magazine, IEEE*, 43(12):112–119, Dec.

[167] W. Richard Stevens. *TCP/IP Illustrated, Vol. 1: The Protocols (Addison-Wesley Professional Computing Series).* Addison-Wesley Professional, us ed edition, 12 1993.

[168] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, IMC '03, pages 39–44, New York, NY, USA, 2003. ACM.

[169] Peter Stuckmann and Rainer Zimmermann. European research on future internet design. *Wireless Commun.*, 16(5):14–22, October 2009.

[170] Nidhi Tare. A Comparative Performance Analysis of GENI Control Framework Aggregates. Master's thesis, Kansas State University, 2010.

[171] J. Touch. Recursive Network Architecture, IEEE Workshop on Computer Communications (CCW). http://www.isi.edu/touch/pubs/ccw2007.pdf, 2007.

[172] J. Touch, Y. Wang, and V. Pingali. A Recursive Network Architecture. Technical report, ISI Technical Report, 2006.

[173] B. Trammell and E. Boschi. An introduction to IP flow information export (IPFIX). *IEEE Communications Magazine*, 49(4):89–95, 2011.

[174] S. Triukose, Z. Wen, A. Derewecki, and M. Rabinovich. Dipzoom: An open ecosystem for network measurements. In *Proceedings of IEEE INFOCOM*. ACM, 2007.

[175] Rodney S. Tucker. The role of optics and electronics in high-capacity routers. *J. Lightwave Technol.*, 24(12):4655–4673, Dec 2006.

[176] Craig Leres Van Jacobson and Steven McCanne. Tcpdump Website. http://www.tcpdump.org/.

[177] Manoj Vellala. Stack Composition for SILO Architecture. Master's thesis, North Carolina State University, 2008.

[178] Verizon Global Latency and Packet Delivery SLA Terms and Conditions. http://www.verizonenterprise.com/terms/global_latency_sla.xml?__ct_return=1.

[179] Verizon IP Latency Statistics. http://www.verizonenterprise.com/about/network/latency/#overview.

[180] AVS-NS3: NS3 Network Simulator Model for Adaptive Video Streaming Research Website. https://github.com/abdo5520/AVS-NS3.

[181] MPEG-4 and H.263 Video Traces for Network Performance Evaluation Website. `http://www2.tkn.tu-berlin.de/research/completed_research.html`.

[182] J. Villalon, P. Cuenca, L. Orozco-Barbosa, Yongho Seok, and T. Turletti. Cross-layer architecture for adaptive video multicast streaming over multirate wireless lans. *Selected Areas in Communications, IEEE Journal on*, 25(4):699–711, 2007.

[183] Anjing Wang. *The SILO Architecture: Exploring Future Internet Design*. PhD thesis, North Carolina State University, 2010.

[184] Anjing Wang and Rudra Dutta. GENI-IMF GEC8 Demo Report. `http://groups.geni.net/geni/wiki/IMF`, 2010.

[185] M.S. Wang, A. Wang, B.G. Bathula, C.P. Lai, I. Baldine, C. Chen, D. Majumder, D. Gurkan, G.N. Rouskas, R. Dutta, and K. Bergman. Demonstration of qos-aware video streaming over a metro-scale optical network using a cross-layer architectural design. In *Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference, 2011 Conference on (OFC/NFOEC)*, pages 1–3, March 2011.

[186] M. West and S. McCann. *RFC 4413, TCP/IP Field Behavior*, 2006.

[187] Carey Williamson. Internet traffic measurement. *IEEE Internet Computing*, 5(6):70–74, November 2001.

[188] R. Winter, J.H. Schiller, N. Nikaein, and C. Bonnet. Crosstalk: cross-layer decision support based on global knowledge. *Communications Magazine, IEEE*, 44(1):93–99, Jan.

[189] Tilman Wolf, James Griffioen, Kenneth L. Calvert, Rudra Dutta, George N. Rouskas, Ilia Baldine, and Anna Nagurney. Choice as a principle in network architecture. *SIGCOMM Comput. Commun. Rev.*, 42(4):105–106, August 2012.

[190] eXpressive Internet Architecture Project Website. `http://www.cs.cmu.edu/~xia/`.

[191] XMPP. `http://xmpp.org/`.

[192] XMPP Publish and Subscribe Extensions Website. `http://xmpp.org/extensions/xep-0060.html`.

[193] Hongyue Zhu, Hui Zang, Keyao Zhu, and Biswanath Mukherjee. A novel generic graph model for traffic grooming in heterogeneous wdm mesh networks. *IEEE/ACM TRANSACTIONS ON NETWORKING*, 11(2):285–299, 2003.

APPENDICES

Appendix A

A.1 The SILO Architecture

In this appendix, we give an overview of SILO Future Internet Architecture and then show our work investigating and comparing different approaches in integrating measurement capabilities into SILO.

A.1.1 The SILO Project Overview

As part of the NSF FIND [51] (Future INternet Design) project, SILO [150] (Services Integrated controL and Optimization) is a clean-slate network architecture enabling dynamically composable per-flow network stacks with a flexible, extensible, cross-layer capable, service-oriented approach as an alternative to the TCP/IP stack. SILO's customized network stacks are composed from fine-grain protocol elements called "services", based on the application requests. While keeping the benefits of layering, cross-layer interaction of these services is allowed through a SILO Tuning Agent (TA) with standard interfaces for optimizing the network performance. Although earlier SILO publications [45, 13, 177] are helpful to understand the concepts, SILO architecture and prototype has evolved, so our work is based on the final prototype which has subtle differences from the earlier architecture. A detailed explanation of SILO can be found in [9, 11, 183] and in this subsection we only give an overview to provide the background to explain our work.

The SILO Architecture: The SILO Architecture aims to introduce a "meta-design" that allows evolution and innovation within the architecture itself for better flexibility and extensibility. In order to achieve this goal, SILO follows a stack-based "generalized layering approach” by using “services” such as “compression”, “encryption” as the building blocks. In SILO, each service is responsible for a particular small function instead of layers, where several functions are embedded into a single layer and it is much harder to make changes. This approach not only preserves the benefits of layering such as data encapsulation and simple buffer management, but also enables easy cross-layer interaction in a standardized way.

In SILO, applications request the services they need, allowing different stacks coexist in different platforms yet using the same architecture. For example, an application running on a sensor node might only require a few simple services whereas a supercomputer might use a different set of services to have high network performance. In Figure A.1 (taken from [183]), a SILO-aware application uses the SILO API to create a SILO socket and request a SILO stack from the SILO Management Agent (SMA), which is the main entity in the architecture. To create a stack, SMA first sends the request to SILO Construction Agent (SCA) to validate (or detect dependencies/conflicts) the request by returning a recipe, called “SILO recipe” which contains a full description of services. SMA then loads the necessary service modules dynamically from the service repository, also called Universe of Services Storage (USS). SMA then returns the stack ID to the user application, so that user application can now start sending data by referring to that stack ID and any data sent, go through each service defined in that stack before leaving SMA.

SILO has built-in support for cross-layer (or cross-service) interactions via the Tuning Agent (STA). Each service may have one or more “tunable” interfaces called “knobs” for cross-service interaction, and therefore STA can query these knobs to get a complete view of that SILO stack and then it can to set new values for these knobs in order to “tune” and optimize the performance. For example, in case of a few consecutive packet losses, knowing the high bit error rate (BER) at the wireless channel, a Tuning Agent can inform the congestion control service about the high BER so that it does not reduce the sending rate, whereas TCP would reduce the sending rate without considering the physical layer conditions. Because different applications might use different Tuning approaches, Tuning Agent itself is only a placeholder to load different Tuning Algorithms (TA) based on again application requests. Thus, it’s the Tuning Algorithm that makes the tuning decisions.

The SILO Prototype: In our experiments, we use SILO version 0.3 [150] released in August 2010 and implemented in C++ and Python on Linux. To keep the implementation simple, SILO is single-threaded and timing based events are scheduled by live555 library [52]. An important timeout based event is the Tuning Algorithm, that is, a Tuning Algorithm is periodically called every N seconds to make decisions and tune service knobs. Communication to and from the destination application is enabled by UDP sockets, i.e. , an application's data first goes through the SILO services and then SMA encapsulates each data packet into a UDP packet to send it to the destination SILO SMA, where the payload goes through SILO services in the opposite order and finally delivered to the destination application. Note that the services at client SMA and server SMA may be different.

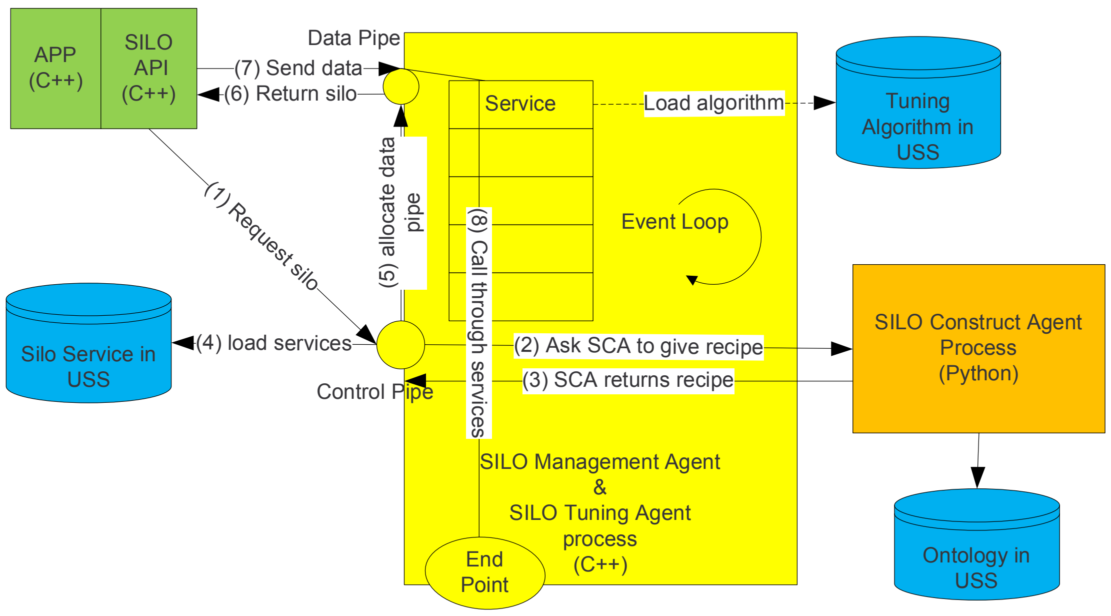


Figure A.1: The SILO Prototype (figure taken from [183])

A.1.2 The Measurement Approaches on SILO

In IMF case study (described in Section 4.1), a SILO measurement service was responsible for subscribing to a XMPP measurement topic and a Tuning Algorithm was querying this SILO service periodically in order to retrieve the measurements about the physical layer conditions. While this was adequate for that case study, we came to realize that we should investigate, which component would be architecturally more efficient to have the measurement consumption functionality. The SILO implementation [150] is single-threaded, and its timing based events are scheduled by live555 library [52] as a queue of events. After several observations, we have noticed that this can potentially affect the quality and effect of cross-layer decisions of the Tuning Algorithm, i.e., a cross-layer decision may not be affective if the measurement data used is not "fresh" enough. In investigating the possible impact of such architectural considerations [11], we compared the performance of existing approach with a new approach where, a Tuning Algorithm directly collects measurement data under various conditions. The main idea was to improve the measurement and actuation feedback loop by finding possible problems. We have setup an example topology with end nodes running SILO stacks where the "source" sends a constant stream of application data to the "destination" and also a Tuning Algorithm (TA) at the "source" periodically gets router buffer and packet loss information as measurement data. We have created two scenarios; 1) a congestion scenario, where TA gets the measurement data in terms of the allocated router buffer size so that it reduces its sending rate to reduce the congestion, and 2) high packet-loss scenario, where TA gets the measurement data in terms of packet loss rate (produced by the intermediate system) and TA does a path switch if the packet loss rate gets too high. Because TA periodically "wakes up" every N milliseconds to consume the measurement data, these experiments are executed for different N values, for both approaches. We have observed unexpected results, because it turned out that the scheduling of TA in a series of events were causing synchronization problems for TA to react to the events especially when it retrieves data from the SILO measurement service, that from now on, we call "service-based" approach vs. TA getting measurement data itself directly, that we call "TA-based" approach. Figure A.2 (taken from [11]) shows how TA-based approach keeps the throughput high by reacting promptly to perform path-switching when BER increases and $\mathsf{N} = 1000ms$ (1 second). Another scenario is the congestion scenario shown in Figure A.3 (taken from [11]), where the "error percentage" is the router's additional (and undesirable) buffer allocation percentage that TA aims to reduce by reducing the sending rate, if this percentage is more than a certain threshold. These differences again are related to the TA's reaction time. For more details about the methodology and results, readers are referred to [11].

From our verification service architecture point of view (discussed in Chapter 3), the node providing the measurement data (BER and router buffer allocation percentage) is

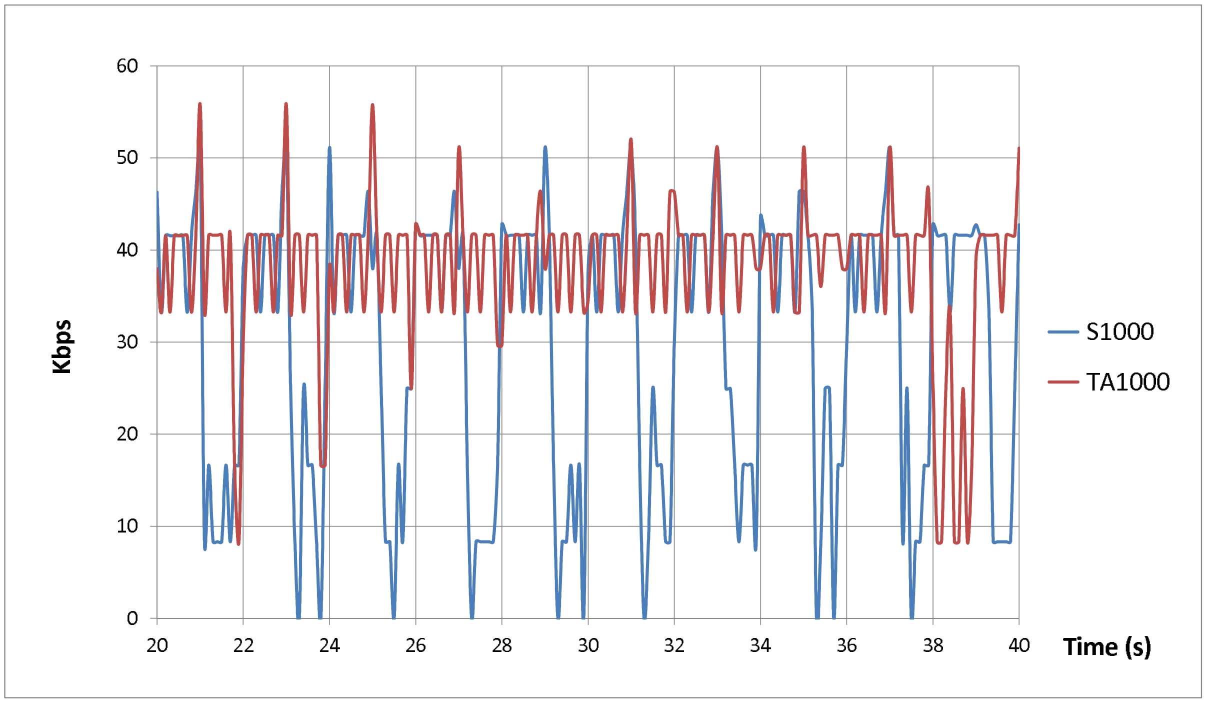


Figure A.2: Throughput of service-based as "S" vs Tuning Algorithm based as "TA" when sleep period is 1 second

the MSP and the analyzer (MASP) is the Tuning Algorithm. Also the application is the chooser and the actuator is the TA reducing the sending rate or switching the path, as part of the "the vote with your wallet" principle, where in the former the chooser wants to pay less and in the latter the chooser wants to switch to an alternative path (provider) with less BER.

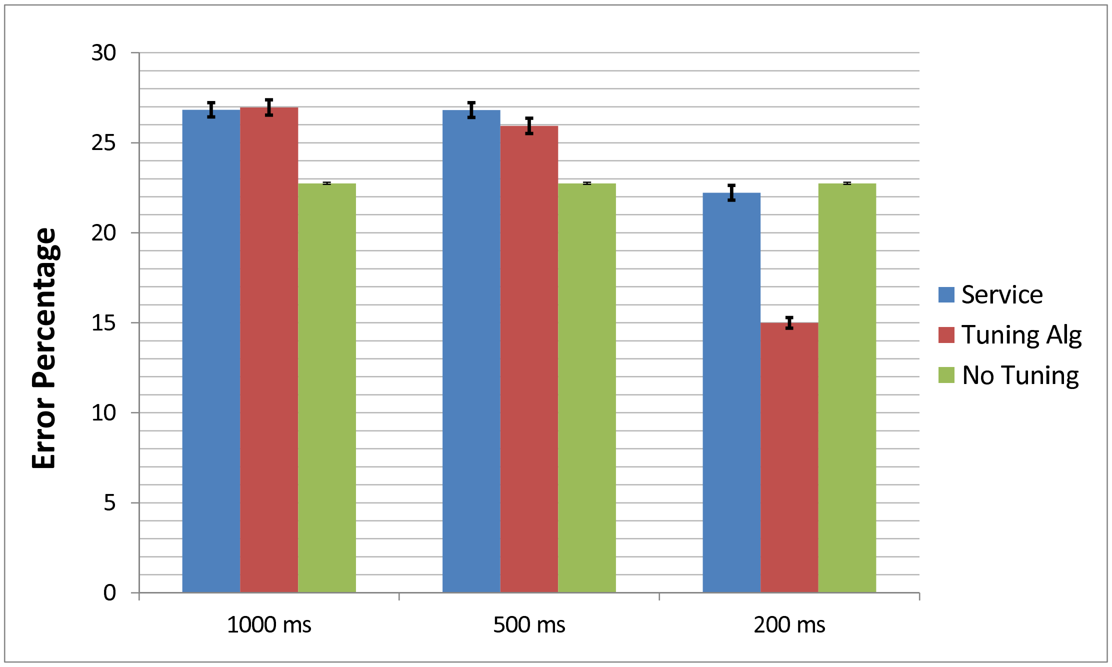


Figure A.3: Error rate as the difference between desired value vs actual value

Appendix B

B.1 GENI

Innovative network protocols and architectures must be throughly tested to convince the community that it is better than the existing procotols and architectures in practice, and so it is worth deploying them. In order to have this testing capability at large-scale, NSF initiated GENI (Global Environment Networking Innovations) project [63] [15], a national-scale, programmable and sharable experimental facility. GENI is still an evolving project, however, it is an excellent opportunity for researchers to run their prototypes on the system and see how they work. We have also contributed to GENI in IMF project (discussed in Section 4.1) and therefore, we would like to give an overview of GENI, experiments in GENI and the control frameworks of GENI.

Overview: The core concepts behind GENI [46] are programmability of nodes, virtualization of resources, federation of different testbeds and easy management of experiments across distributed systems.

Experiments in GENI: A researcher who wants to make an experiment, first requests a "slice" from the clearinghouse using his/her credentials obtained from GENI Authorities. A slice is a set of resources which may be owned by different authorities, called "aggregate managers". These resources can include, but not limited to, virtual machines, virtual CPUs, memory, handheld devices, routers, programmable core switches or fiber-optic backbone links. When a request comes to the clearinghouse, it first checks the user credentials. It, then runs the control framework, which reserves the requested "slice" with the specified resources in it, by negotiating with the aggregate managers. Now that the slice has been reserved, the experimenter can run its experiment on this slice.

Control Frameworks: The core component of GENI, a GENI control framework can be divided into four planes [170]: control plane, data plane, operations & management plane and measurement plane. The control plane [116] controls the creation, maintenance and removal of slices. Control frameworks typically compose slices by stitching several computer and network resources together in GENI. The data plane carries the experimental data. The operations and management plane checks the experiment validity to detect security threats and misbehaving slices. The measurement plane provides the capability of measuring the traffic within the slice nodes [67, 16] or in some cases the physical layer substrates (such as our IMF project described in Section 4.1). Currently GENI has four control frameworks; PlanetLab, ProtoGENI, ORCA and ORBIT [63, 116]. Additionally, a popular technology that came with GENI is Openflow as we discussed in Section 2.2.1.

Current Status: GENI is now at a useable status with its user-friendly web-browser interfaces such as Flack [56]. However higher reliability and addition of new features (such as measurement features) are still an ongoing effort. GENI has also proved to be an excellent tool for teaching purposes as we have used GENI in our graduate-level networking courses and projects. A possible interesting outcome of GENI experimental facility is that GENI itself can become the future Internet Architecture [47], i.e., virtualized networks having different capabilities, including the existing Internet, may simultaneously run on GENI, making GENI the future Internet Architecture. In this thesis, we only refer GENI as the testbed that experiments run on.