A SOFTWARE-DEFINED NETWORKING APPROACH FOR WIRELESS SYSTEMS

A Dissertation

by

Volkan Yazıcı

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in the
Department of Computer Engineering

Özyeğin University
May 2015

A SOFTWARE-DEFINED NETWORKING APPROACH FOR WIRELESS SYSTEMS

Approved by:

Assoc. Prof. M. Oğuz Sunay, Advisor
Department of Electrical &
Electronics Engineering
Özyeğin University

Prof. Murat Tekalp
Department of Electrical and
Electronics Engineering
Koċ University

Prof. M. Reha Civanlar
Department of Electrical &
Electronics Engineering
Özyeğin University

Assist. Prof. Müge Sayit
International Computer Institute
Ege University

Date Approved: 8 May 2015

Assist. Prof. Ali Özer Ercan
Department of Electrical &
Electronics Engineering
Özyeğin University

ABSTRACT

The tremendous growth in wireless Internet use is showing no signs of slowing down. Existing cellular networks are starting to be insufficient in meeting this demand in part due to their inflexible and expensive equipment as well as complex and non-agile control plane. Software-defined networking (SDN) is emerging as a natural solution for the next generation cellular networks as it enables further Network Functions Virtualization (NFV) opportunities and network programmability. In this dissertation, we advocate an all-SDN network architecture with hierarchical network control capabilities to allow for different grades of performance and complexity in offering core network services and provide service differentiation for 5G systems. As a showcase of this architecture, we first introduce a unified approach to mobility, handoff and routing management and offer Connectivity Management as a Service (CMaaS). CMaaS is offered to application developers and over-the-top service providers to provide a range of options in protecting their flows against subscriber mobility at different price levels. Next, we present the implementation details of a distributed SDN controller specifically crafted to realize the proposed all-SDN architecture and investigate the flow-level performance characteristics of the system.

ÖZETÇE

Gün geçtikçe artan internet kullanım oranı hızını kesmeden büyümeye devam etmektedir. Pahalı ve karmaşık olmanın yanı sıra gerekli esneklik ve çeviklikten uzak olan mevcut hücresel ağlar ise bu artan talebi karşılamakta yetersiz kalmaktadır. Sağladıkları Ağ Fonksiyonu Sanallaştırması ve programlanabilirlik kabiliyetleri ile Yazılıma Dayalı Ağlar (YDA) yeni nesil hücresel ağlar için doğal bir çözüm olarak karşımıza çıkmaktadırlar. Bu doktora tezinde, merkezi ağ servislerinde farklı başarım ve karmaşıklık seviyelerine ve 5G sistemlerde servis farklılaştırılmasına imkan sunan, hiyerarşik ağ kontrol donanımına sahip bütün bir YDA ağ mimarisi yaklaşımı savunulmaktadır. Bu mimarının somut bir temsili olarak ilk önce hareketlilik, hücreler arası transfer ve rotalama yönetimini birleşik bir şekilde ele alan Servis Olarak Bağlantı Yönetimi (SOBY) yaklaşımı tanıtılmaktadır. SOBY, uygulama geliştiricileri ve servis sağlayıcıları için, üyelerin hareketliliğinin ağ üzerindeki akışlarına olan etkisinin ücretlendirmeye göre farklılaştırılmasına olanak sağlamaktadır. Devamında, bahsi geçen YDA mimarisi için özel olarak geliştirilmiş yazılımsal yönetici gerçeklemesi ayrıntılandırılarak, sistemin akış başarım karakteristiği incelenmektedir.

ACKNOWLEDGEMENTS

I would like to thank to the Scientific and Technological Research Council of Turkey (TUBİTAK BIDEB 2211-C), for its financial support during my PhD process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Mobile operators are faced with several major challenges such as the unprecedented increase in wireless traffic, losing existing and new revenue sources to over-the-top (OTT) providers, and increasing capital as well as operational expenditures to serve the demand. To meet these challenges, many mobile service providers (MSPs) are heavily pursuing cloudification opportunities, mainly in the form of Network Function Virtualization (NFV) [1]. NFV aims to move some or all of the functions of the mobile network from dedicated hardware platforms to virtual machines running on generic hardware. NFV promises reduced expenditures, agility and flexibility and the capability for MSPs to launch new network services to seize new market opportunities at time scales that OTT providers can achieve. Virtualization and pooling of baseband processing in the base stations is one example of NFV, and is commonly referred to as the Cloud-RAN (C-RAN) in the literature [2].

Moving towards cloudification, a brute-force approach would be to keep the current network architecture and simply run core network nodes (e.g., xGSN, S-GW, P-GW, MME, PCRF, HSS, etc.) as well as service platforms (e.g., IMS) in a virtualized data-center environment (referred to as Telco Cloud) [3]. We argue that while this is a step in the right direction towards easing the problems of the MSPs, it is not sufficient. Ultimately, a mobile network is concerned with the forwarding of flows to and from mobile user equipment (UE) via a chain of network functions. While NFV brings programmability, agility and flexibility to the realization of individual functions, it is essential that a programmable, agile and flexible realization

of individualized flow control that orchestrates flows across different chains of functions is also present in 5G systems so that MSPs can quickly create and deploy new, revenue-generating services.

We advocate that a key differentiator of 5G systems from 4G will be in how we architect and orchestrate the overall system control to realize the benefits of cloudification while taking the full advantage of the transport capacity distributed over a large geographical area in the form of base stations, switches, routers, fiber links, microwave links, etc. We envision fully decoupled, independently scalable and programmable user and control planes for 5G. In this respect, Software Defined Networking (SDN) is a natural architectural choice for 5G systems. In SDN architectures, complex control plane functions (CPFs) are removed from forwarding elements and placed behind a logically centralized controller. Thus, SDN approach simplifies the forwarding elements and ships complex CPFs to physical or virtual servers running in a data-center. The controller collects the distributed network state on behalf of CPFs and provides them a direct centralized access to raw network state or their abstracted forms. Based on this centralized view, CPFs dictate decisions on how packets are processed, pipelined, and forwarded on one or more data plane nodes (DPNs) by sending instructions back to the controller, which in return sanitizes and translates these instructions for individual forwarding elements using an open standard (e.g., OpenFlow, NetConf, etc.) or proprietary interface supported at individual DPNs.

Mobile networks are composed of two components: the Radio Access Network (RAN) and the Core Network (CN). While RAN provides connectivity of the UEs to the network via base stations (eNBs), the CN provides paths between eNBs and various services as well as outside networks. Then, considering the mobility of UEs, the delay constraints associated with various control functionalities of the mobile network are significantly different. In this dissertation, while we advocate an all-SDN network architecture for 5G, we ascertain that an inter-working set of hierarchical controllers

as opposed to a single centralized controller is necessary to handle such variations in the delay constraints. The hierarchy of controllers allow for not only locally optimized control decisions within the network, but also a new dimension in service provisioning, where control of a given service at different hierarchies corresponds to different grades of service.

1.1 Why The 4G Control Plane Needs To Change

The 4G cellular system has evolved from its third generation counterpart. For this reason, the RAN and CN components are usually referred to as Long Term Evolution (LTE) and System Architecture Evolution (SAE), respectively. Together, they form the Evolved Packet System (EPS) [4]. EPS is an all-IP network supporting only packet-switched connectivity. All radio related functions are pushed down to the eNBs in the EPS to increase delay performance when reacting to the changes in the wireless environment. EPS has a clean separation of the user and data plane in the CN to allow the independent scaling of the two planes.

The control operations in the EPS are functionally split between the RAN and the CN. The RAN has a single element, namely the eNB. The eNB is responsible for admission control; inter-cell radio resource management (RRM); radio resource block (RB) control and scheduling; and handoff management. In contrast, the CN consists of several elements, namely Mobility Management Entity (MME), Serving Gateway (S-GW), Packet Gateway (P-GW), Home Subscriber Server (HSS), and Policy & Charging Rules Function (PCRF). MME is mainly responsible for paging and mobility management, but is also involved in bearer management, admission control, subscription management, etc. HSS is involved in subscription management. PCRF, S-GW and P-GW are all involved in bearer establishment, maintenance, and quality of service (QoS). For instance, PCRF dictates QoS policies and charging for individual flows and subscribers, while enforcement of these policies through mapping

flows to bearers, performing packet filtering and metering are tasks of the P-GW. P-GW is also responsible for IP address assignment for the UEs. S-GW is involved in buffering packets for idle mode subscribers and triggering paging through MME. The current EPS network architecture is illustrated in Figure 1.

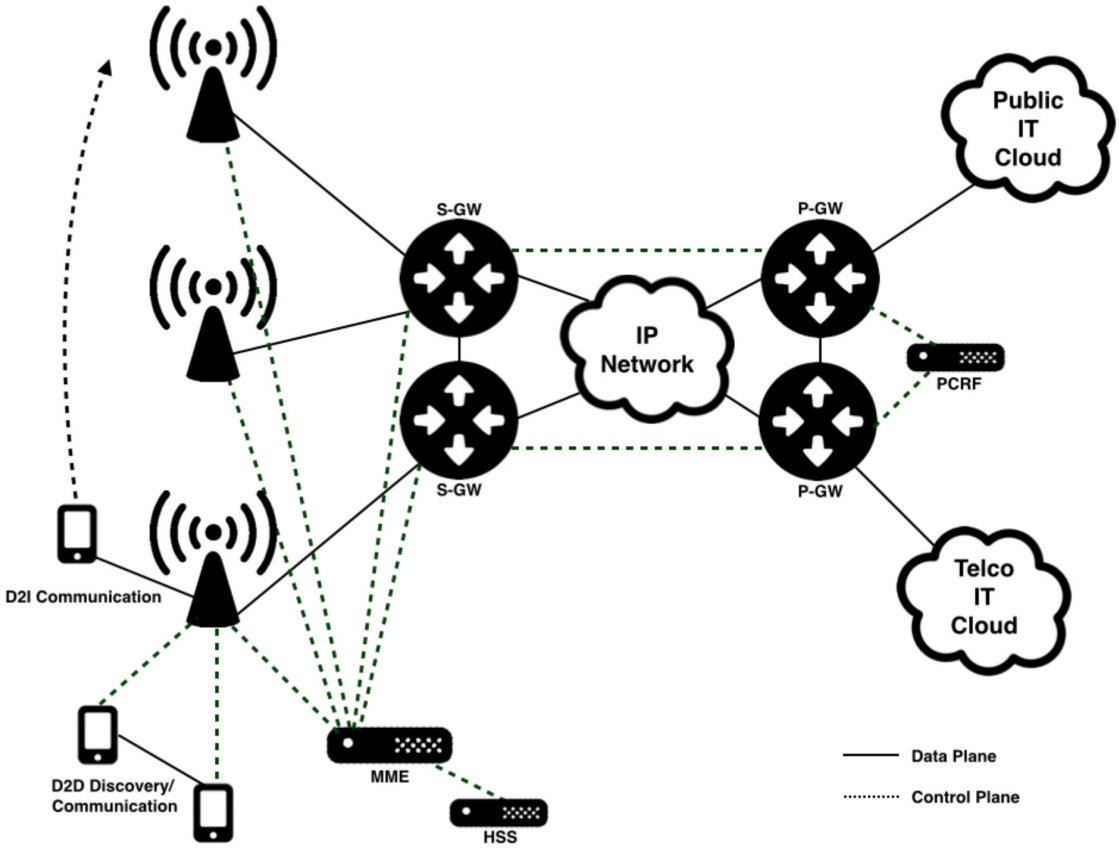


Figure 1: Current EPS Network Architecture

The massive growth in the traffic volume as well as the volume of connected devices necessitates an evolution of the LTE towards 5G. The latest significant step in this evolution is 3GPP Release 12, where the new use case of device-to-device (D2D) communication is being studied for the first time under the title "Proximity Services (ProSe)" [5]. 3GPP defines D2D communication as the communication between two nearby UEs directly, without routing through the EPC. The D2D communication may or may not involve routing through the local eNB. D2D communication will improve spectrum and energy efficiency of the overall system while increasing the throughput

and end-to-end delay performance for the D2D links [6]. D2D-enabled LTE is also aimed for mission critical communication systems that must function when cellular networks are not operational.

In 3GPP, D2D studies are centered around two fundamental types of operations: (1) D2D discovery (2); and D2D communication. While the D2D discovery aims to identify other UEs in close proximity for possible D2D communication, D2D communication is the actual direct link for data transfer. For both operations, it is essential to architect an efficient, fast and robust control plane, so that D2D terminals are time synchronized to the network, are allocated and subsequently scheduled the necessary resources.

D2D discovery should be designed in such a way that the UEs wake up only to listen for potential D2D partners. This is best achieved by allocating static wireless resources to the discovery operation in the network. Conversely, the D2D communication should utilize resources only when necessary so that spectrum efficiency is maintained in the network. This requires a dynamic, network-controlled allocation of resources to the D2D link. For both D2D discovery and communication, the uplink resources are favored [6].

D2D discovery and communication operations should be operational beyond the coverage of the network. In the absence of the network guidance, the pool of UEs that are candidates for D2D communications may either use an ad-hoc or cluster head-based control mechanism. If an ad hoc mechanism is employed, each UE controls itself and discovery and communication may utilize one of the well-known random MAC protocols such as CSMA [6]. If, on the other hand, a cluster head-based control topology is preferred, one UE assumes a master role and performs all of the control operations that local eNBs provide under network coverage. The cluster head-based control is also suitable for D2D-based range extension within the network, where the selected cluster-head acts as a relay to convey network control and communication

data to out-of-coverage UEs that are in its proximity.

While LTE has achieved significant gains in lowering the MSP expenditures and increasing end-user experience with higher data rates and lower latencies, a new thinking of the cellular network architecture is necessary towards 5G. The explosive demand in wireless data is driving a heterogeneous network (HetNet) paradigm having a large number of small cells as well as allowing D2D connectivity and resulting in potentially a significantly more complex RAN and CN control, pushing up the deployment and operational costs substantially. 5G networks will likely experience a continuous deployment of small cells based on changes in local user demand. As it is not possible to re-architect the network every time it gets denser, an agile and configurable solution is needed. Furthermore, it is very likely that the local demand for the data and control planes of 5G networks will grow at different paces, necessitating an independently scalable solution. As the user demand increases, so does the user intolerance to under-performing applications. 5G networks should be able dynamically steer or re-prioritize individual traffic flows based on network-wide orchestration that utilize big data analytics to ensure user satisfaction. For improved performance, 5G networks will also require a more coordinated approach to RAN technologies, as already exemplified in the latest release LTE systems in the form of coordinated multi-point (CoMP) transmission and inter-cell interference coordination (ICIC) where multiple eNBs serve a UE in coordination and multiple eNBs coordinate their transmissions to minimize their interference to one another, respectively. Such coordinated technologies will potentially be better realized via a logically centralized control plane spanning over the eNBs of a given geography as opposed to the current distributed approach of LTE. Furthermore, reduction of the costs of such a network will need to come in two major fronts:

1. **Cloudification:** Virtualization of various network functions will enable the

realization of a multitude of functionalities in a virtualized data-center environment, eliminating the need for specialized hardware;

2. **Programmability:** A programmable network will enable MSP-led innovations of new control applications and corresponding chain of services to provide control differentiation of different flows for even some of the most fundamental network operations such as mobility management.

Then, an agile and flexible 5G architecture with perfectly decoupled data and control planes where virtualized network functions as well as data flows can be orchestrated programmatically is necessary. We argue that the SDN framework is the ideal candidate for such an architecture. After providing a brief review of the existing SDN proposals for cellular networks, we introduce the proposed architecture.

1.2 Overview of Existing SDN Proposals for CN and RAN

There have been a number of studies in the literature that detail how the current 3GPP data plane may be realized using an SDN framework. For instance, in [3], Kempf et al. propose to support the tunneling protocol between the gateway nodes and eNBs as an extension to OpenFlow Switch (OFS) Specification [7]. Furthermore, they envision that the transport fabrics in between the tunnel end points are also OFS, centralizing the control over the paths that flows follow between the P-GW and eNBs. As a result, both S-GW and P-GW can be split from their data plane functions and instead run purely as control plane applications. The proposed architecture however does not mention how complex policies can be enforced on the extended OFS, a crucial component for carrier-grade SDN. MobileFlow introduces a complementary SDN architecture with a logically centralized MobileFlow controller for mobility management with legacy equipment support in addition to the OpenFlow controller for routing [8]. While MobileFlow provides a promising architecture, it does not specify the interplay of the two controllers in detail for different network control

operations, nor does it mention how the control of the RAN might be integrated into the architecture.

The Softcell architecture highlights possible bottleneck issues due to many functions hosted on P-GWs and advocates a cleaner separation between the data and control plane functions [9]. In the proposal, local control agents interact with the more centralized controller to resolve the time-scale issues in control loops. Authors advocate that the current OpenFlow model is not sufficient to perform useful functions such as DPI and header compression on the path.

The OpenRoads architecture discusses the necessity of coordination between the RANs of different radio access technologies (RATs) to enable seamless inter-RAT handoffs [10]. OpenRadio discusses the benefits of a software programmable data plane for the RAN through the decomposition of the wireless protocols into separate processing and decision plane components with a simple, programmable API between them [11]. SoftRAN suggests a logically centralized control plane for the RAN in a given geography composed of a number of eNBs, where parts of the control remain at eNBs [12]. Specifically, operations of handoffs and power control are handled at a centralized network controller whereas each eNB controls resource allocation. OpenRF proposes a central controller for coordinated interference management of MIMO-based Wi-Fi networks [13]. The OpenRF controller assigns each flow to an access point (AP) and establishes corresponding interference and coherence vectors. These assignments are conveyed to the APs, which in return combine all assigned coherence and interference vectors to produce precoding vectors that enable transmission of desired flows coherently while nulling any interference that these flows may cause to other active flows.

Extending on all of the above reported work, we propose an all-SDN architecture for the mobile network with hierarchial controllers. In the next section we discuss this architecture.

1.3 The Contribution

The main contributions of this dissertation are as follows:

1. Proposal of an all-SDN network architecture with hierarchical network control capabilities to allow for different grades of performance and complexity in offering core network services and provide service differentiation for 5G systems.

2. Introduction of the Connectivity Management as a Service (CMaaS) concept, where a unified approach to mobility, handoff and routing management is described.

3. Evaluation of cellular connectivity management schemes under infrastructure-to-device and device-to-device communication scenarios.

4. Proposal of new CMaaS schemes considering latency, throughput, and infrastructure load trade-offs between different reactivity models.

5. Evaluation of the flow-level performance characteristics of the distributed controllers in the proposed all-SDN architecture with an exhaustive set of configurations.

6. Implementation of an IGMP-based synchronization and messaging protocol between a multitude of hierarchical network controllers in order to realize the proposed architecture.

The rest of this dissertation is organized as follows: In Chapter 2, an outline of the the proposed all-SDN network architecture is presented. Next, in Chapter 2, CMaaS is described in detail. Here Sections 3.1 and 3.2 explain a unified programmable handoff and routing control for device-to-infrastructure and device-to-device links, respectively. And Section 3.3 shares the performance comparison of various approaches. In

Chapter 3.4, flow-level performance characteristics in multi-domain SDNs are investigated and evaluated. Sections 4.2 and 4.3 presents the proposed distributed controller architecture and the employed experimental setup, respectively. Section 4.4 discusses the obtained results. Additionally, in Section 4.5 a thorough study of the related work in the literature is presented. Finally, in Chapter 5 ends the dissertation with a conclusion.

CHAPTER II

A NEW PROGRAMMABLE 5G CONTROL PLANE ARCHITECTURE

The 5G network will most likely be heterogeneous in its deployment with densely populated small cells, with device-to-infrastructure (D2I) as well as device-to-device (D2D) links and operational on a number of different carrier frequencies, ranging from today's cellular bands below 5 GHz to millimeter-waves at 60 GHz and beyond. Evolving from LTE, 5G will likely have an extensive set of adaptive physical layer components relying on large numbers of transmit and receive antennas.

We believe a simpler, programmable network architecture will be able to support such a vision. This architecture will completely eliminate specialized and thereby expensive components such as MME, S-GW, P-GW and PCRF as well as tunneling protocols used for overlay routing. Furthermore, this architecture will unify the control of the RAN and the CN for the network to allow for flexibility in orchestrated network programmability. While we advocate an all-SDN network architecture, we ascertain that an inter-working set of hierarchical controllers as opposed to a single centralized controller is necessary to handle the delay constraints associated with various control functionalities of the mobile network. For instance, scheduling of wireless resources is traditionally based on the channel quality feedback received from the users. The coherence time of this feedback is dependent on the carrier frequency as well as the user mobility. At 2 GHz, this is equal to 90 ms and 1.1 ms for a user travelling at 3 km/h and 250 km/h, respectively. At 5 GHz, these values reduce to 36 ms and 0.43 ms, respectively. Considering a 5-10 ms one-way delay in traditional backhaul links [12], a centralized control of such a scheduling operation away from

the individual eNBs may be feasible for a low mobility user, but certainly not for a high mobility user. The proposed architecture provides the flexibility to divide the wireless resources within a geographical area with multiple eNBs into a number of virtual slices and perform scheduling within these slices at different controller hierarchies. While scheduling at the eNB Controller allows for reactive utilization of wireless resources, it is conducted with a limited, local network view. On the other hand, scheduling conducted at the RAN Controller may lead to better prioritization of flows, utilizing a larger network view, at the expense of an increase in the reaction time to network dynamics.

In the proposed architecture, which is depicted in Figure 2, the core control functions of the wireless network such as connectivity, radio access technology (RAT) selection, handoff management, mobility management (MM), coordinated radio resource management (C-RRM), QoS, policy and charging are all realized as applications running on one or more of the hierarchical controllers. Furthermore, multiple control applications for the same functionality may be present in the network, realized at the same or different controller hierarchies. The selection of the control application for a given flow may depend on not only the user identity and flow and connection type, but also user mobility, user observed channel quality, network carrier frequency, mobile phone capability, user billing plan, roaming information, OTT identity, etc. that jointly make up the mobile network state. It may also be possible to switch from one control application to another during the lifetime of a flow due to a change in the network state.

As illustrated in Figure 2, the following controllers are defined in the proposed architecture in decreasing hierarchy: Network Controller, RAN Controller, Base Station (BS) Controller and User Equipment (UE) Controller. For a mobile network covering a wide-area, it may be desirable to realize the logically centralized Network

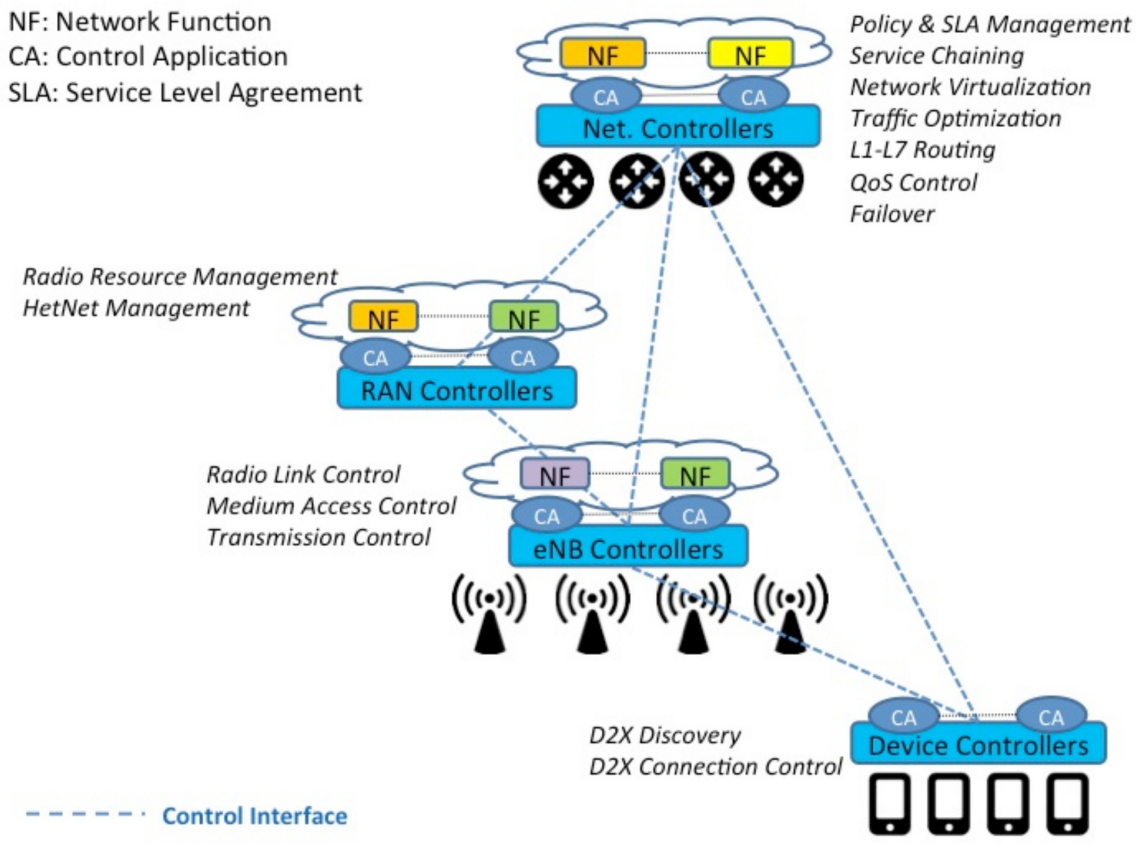


Figure 2: Programmable All-SDN 5G Network Architecture

and RAN Controllers in a distributed fashion across the topology to provide scalability and increased performance. By their nature, the BS and UE Controllers are already distributed.

Analogous to the OpenFlow interface [7], the control of the end-to-end network operates over tables of `<Match, Action>` tuples, sent by the controllers to the forwarders (routers, base stations and user equipment). However, an extended set of match and action attributes are needed for the mobile network. Any subset of the above described network state may be used as a Match entry. The corresponding Actions may be to select RAT, schedule or avoid a specific wireless resource, initiate handoff, set the modulation and coding, initiate an ARQ protocol, charge according to a specific policy, initiate CoMP, initiate ICIC, forward on a specific port, pause

a flow, resume a flow, limit the data rate and bandwidth, allow/disallow D2D communication, act as a relay, etc. An efficient representation of the network state and associated control actions within the `<Match, Action>` tuple is necessary and requires further investigation.

The control of the network also necessitates close interaction between the hierarchical controllers. In the proposed architecture, a controller at a higher hierarchy may send constraints to a controller at the lower hierarchy using a similar `<Match, Action>` tuple. Here, the `Match` entries may include any subset of the network state and the `Action` entries include the selection of a control application, disabling or enforcing the joining of a RAT, limiting modulation and coding options, disallowing simultaneous scheduling of the same resource to the matching flows, disallowing handoffs to certain types of base stations, disallowing or only allowing D2D communication, limiting transmit power for a given resource, powering on or off a base station, etc. Conversely, a controller at a lower hierarchy sends abstracted feedback to a controller at the higher hierarchy upon the higher hierarchy controller's demand or in regular intervals. For example, the RAN Controller may disallow the handoff of a high mobility user from a macrocell to a small cell. Similarly, the RAN Controller may require the same resource blocks to be or not to be scheduled for a given user by the BS Controllers for CoMP operation. This constraint does not however negate the autonomous operation of the scheduler application running on the BS Controllers. In return, these controllers may send an abstracted feedback of the users' observed channel quality histories to the RAN Controller, which in turn may use this information to decide when and how to instantiate CoMP or ICIC.

Let us now describe the controllers in the hierarchy. The UE Controller is responsible for the selection of one of many available radio access technologies (RAT) that the device supports subject to the limitations that are imposed locally or by one of the controllers in the higher hierarchy. This way, RAT selection based on subscription,

policy and charging on a per-flow basis becomes possible. The UE controller is also responsible for various D2D discovery and communication control operations: push or pull based discovery control, D2D physical layer modulation and coding adaptation, H-ARQ operation, out-of-coverage D2D distributed random access and/or cluster head-based centralized control options for discovery and communication, etc.

One step higher in the control hierarchy is the BS Controller. As described above, we believe that delay-constrained functions such as wireless resource management and scheduling as well as the corresponding adaptive physical layer packet creation need to be controlled close to the UE at the BS Controller for D2I communication. The D2D resource management and synchronization of the in-coverage D2D UEs are also controller by the BS controller. The RAN Controller, which is one step up on the hierarchy, oversees all of the BSs in a given geographical area and thus has the potential to effectively control the C-RRM functionalities. However, the proposed architecture also allows for the C-RRM control at the BS Controller in a distributed fashion, similar to today's LTE solution. It is also possible to invoke one C-RRM control for one flow, and another control for another.

The Network Controller at the top of the hierarchy potentially orchestrates end-to-end QoS provisioning, application-aware route establishment and service chaining, mobility management, policy and charging and it percolates/delegates its decisions on the controllers in the lower levels of the hierarchy.

The network controller at each level of the hierarchy is set to be the logically central place for orchestration of the lower level controllers and forwarding equipments. Nevertheless, there are two major drivers that require a logically centralized, but physically distributed controller implementation. First, the physical network components are inherintly spread over a vast geographical area. Second, stringent scalability and reliability measures are necessitated by the underlying communication standards. We advocate a distributed controller implementation architecture such that the flow-level

performance characteristics (flow setup time, data latency and throughput, etc.) are subject to a set of detailed evaluations to meet the aforementioned measures. In our model, we realize such a controller implementation using OpenFlow as a south-bound protocol and IP IGMPv3 [29] for inter-controller messaging.

The hierarchical control of the all-SDN network allows for the realization of a given control operation at different hierarchies, possibly using different control applications and introducing new venues for revenue generation for MSPs. One important example of this flexibility is for connectivity management, which is comprised of mobility management and dynamic route management. This is discussed in the next chapter.

CHAPTER III

JOINT CONTROL OF ACTIVE MODE MOBILITY AND ROUTE MANAGEMENT

One of the fundamental goals of network control is to ensure that a route for a given flow between two nodes is quickly and effectively established and maintained. In a wireless network, this problem becomes significantly more complex as one or both of the nodes are mobile. For D2I links, the control of mobility is split in LTE. A typical handoff is controlled at the RAN. In cases where anchor points (S-GW, P-GW) in the core network alter as a result of the handoff, MME, S-GW and potentially P-GW can be involved to maintain the overlay routes (e.g., GTP tunnels). The selection of IP routes between e-NB, S-GW, and P-GW are controlled via IP- routing independently from mobility management. Furthermore, the MSPs have no choice but use the control functionalities for these operations provided to them in their specialized hardware.

For D2D links, no mobility control operations have been defined in LTE yet. However, seamless connectivity needs to be maintained when the control plane for one or both of the UEs go through a handoff, or when the D2D link is no longer feasible and has to be switched to a pair of D2I links.

In the proposed network architecture, the MSPs will have the option to conduct handoff and route management together and deploy different control applications for this purpose for different flows or users. We refer to this new paradigm as *Connectivity Management as a Service*.

3.1 Connectivity Management as a Service (CMaaS)

Legacy 3G/4G systems carry on the notion of providing almost lossless and low-delay handoffs between neighboring base stations. Such a strict handoff management might be desirable for a high quality (paid) VoIP service, but requiring mobile operators to provide the same stringent delivery performance for all flow types and non-paying OTT services is an expensive proposal as the connection and mobility management is pushed deeper into the Telco Cloud for denser small cell deployments. With the wide adaptation of new protocols such as DASH and using upper layer solutions that already have intelligence in maintaining session connectivity using new transport layer solutions (e.g., multipath TCP) or cloud based connection management, most signaling overhead due to handoff management may be offloaded to third party services for applications without stringent delay constraints. The proposed SDN-based 5G architecture, via the deployment of CPFs at different controller hierarchies, allows service differentiation at the level of connectivity management in a programmatic way for some network flows to achieve much higher performance as their service grades increase. This in return results in the connectivity management as a service offering and may be utilized by the MSP in one of three ways:

1. For services operated by the MSP itself, a connectivity management CPF that has a higher operational cost is used only for flows that require the associated stringent delay and packet loss levels. For others, an operationally cheaper CPF alternative is invoked;

2. A paying user may always be served by the higher grade connectivity management CPF regardless of its flow type;

3. For services operated by OTTs, the higher-grade connectivity management CPF is invoked only to paying OTTs.

Then, CMaaS allows the MSP to lower its operational cost without sacrificing the quality of service for its own applications while introducing new revenue paths.

3.2 Unified Programmable Handoff and Routing Control for D2I links

The all-SDN 5G-network architecture allows a unified control of handoff and routing by jointly using controllers in the same or different hierarchies.

One possible realization of such unified control may be conducted at the Network Controller. In this case, for flows that are to be controlled by this grade level, each BS Controller sends relevant feedback in the form of observed channel quality levels to the Network Controller, which in turn decides when a handoff occurs. This decision is pushed down to the BS Controllers and simultaneously triggers the associated route update between the UE's new position and the egress node. For delay sensitive flows, such a realization may not be desirable. However, handoffs intended for load balancing and/or user mobility handling may benefit from such a centralized approach. An MSP may decide to turn off a given base station to save energy and handoff all its users to neighboring base stations using this approach. Additionally, handoff decisions that might result in significant congestions in a given part of the CN may be avoided thanks to such a unified approach.

One alternative realization is to keep the handoff control at the BS and RAN Controllers but in close coordination with the routing control that is realized at the Network Controller. In this case, a handoff triggers a routing and location tracking update. Depending on the desired service grade, this update may be reactive or proactive. When a reactive update is used, the Network Controller establishes a new route for the flow only after a handoff occurs. The flows in both directions are paused until the new route is established. Alternatively, using a proactive update, every time a handoff occurs, the Network Controller formulates a priori routing decisions for a given flow for all candidate next handoff locations that may be decided intelligently

using user's current mobility path and/or long-term mobility behavior to the egress node. This procedure should also be invoked at the set up of a new flow. Depending on whether there is a direct link between the source and target eNB, the proactive handoff may involve either only the corresponding eNB controllers or additionally the RAN controller. Representative Message Flows for the proactive and reactive mobility control operations are given in Figure 3. The execution of either of these approaches as a CMaaS requires extensions to OpenFlow. Details of such an execution are discussed next section after discussing how handoff control is accomplished for D2D links in the proposed architecture.

3.3 Programmable Handoff Control for D2D Links

A D2D link has its data plane between the two UEs. However, the control plane of the link is between a controlling eNB (or a cluster head UE) and the UEs. Thus, the UE mobility potentially affects one or both of the control planes and/or the data plane. In this regard, as illustrated in Figure 4, three types of handoffs may take place for a D2D link: (1) Single Mobility: The control plane of one of the UEs may be handed off to a different eNB, (2) Dual Mobility: The control planes of both of the UEs may be handed off to one or two new eNBs, (3) D2D to D2I switching: The D2D link may become unsuitable for communication and thus data plane needs to be handed off to two D2I links, one per UE. The proposed all-SDN network control architecture supports all three types of D2D handoffs.

To aid in the handoff decisions, the D2D link as well as the control plane link states need to be regularly fed back by the UEs to the controlling eNB [14]. We argue that for all three cases, only reactive handoffs are feasible as a proactive handoff control would result in a very inefficient use of the wireless resources. The handoff procedure for the first two cases involves the UE controllers as well as the source and destination BS controllers. When the source eNB decides on a control plane handoff,

it forwards this decision to the RAN controller, which in turn coordinates the resource management for the two eNBs for the D2D link. In the case of dual mobility where the destination control plane eNB is the same for both UEs, the handoff control is handled by the two involved BS controllers and the RAN controller is not included since no inter-cell coordinated resource management is necessary to sustain the D2D link. The D2D to D2I switching may potentially include the RAN and the Network controllers as well as the involved BS controllers, depending on how the two eNBs are to be linked to form the new route between the UEs.

3.4 Performance Comparison of Reactive and Proactive CMaaS

We now investigate the performances of reactive and proactive CMaaS for D2I links and the reactive CMaaS for the D2D links in the proposed architecture to highlight the interplay between the data plane performance and control plane complexity in the proposed architecture. We investigate D2I and D2D cases separately and consider a simple cellular network map with 1 GHz links as illustrated in Figure 5 for a geographical area of 8 eNBs. We investigate CMaaS using Mininet 2.1.0 and the Floodlight controller. The evaluated experiments consider three different user populations: 100, 250 and 500. For the D2I mobility experiment, we assume that UEs are uniformly distributed across the eNBs at the beginning of the experiments. For the D2D mobility experiment, we further pair the UEs within each eNB to form direct links. Each UE is assumed to generate TCP flows randomly to one of the two available services for the D2I experiment, and to its pair UE for the D2D experiment, respectively, following an exponential distribution with parameter 1, where the flow durations are uniformly distributed between 0 and 10 seconds. Following this model, a UE may generate multiple parallel flows for a given duration of time. The sources in the network are assumed to generate flows with average data rates of 66 Mbps.

100 30-minute experiments have been conducted for reactive and proactive CMaaS

in this set-up. For the D2I experiment, in the reactive mode, after a handoff occurrs, the controller is queried and a new route is computed towards the corresponding service and the necessary set of rules are pushed down to all relevant network nodes. In the proactive mode, the routes for all possible candidate handoff nodes are computed a priori so that routes are ready prior to the next handoff. While this approach enables an almost zero handoff overhead on the network, computation and establishment of the a priori routes necessitate a certain overhead. In order to address this problem, an alternative approach is introduced such that a randomly chosen half of the routes are established. In the experiments we assume that each eNB has 6 neighbors that are candidates for the next handoff. Even though the proactive mode prepares routes for the next handoff in advance, flows may still experience slight delays due to control plane operations. This is due to the expiration threshold of flow rules and the fact that the controller does not retransmit an already active rule to a switch and eNB, resulting in a slight possibility that the proactive CMaaS may contain reactive components. A brief summary of evaluated reactive and proactive approaches can be given as follows:

- **Reactive & Proactive (OpenFlow):** Standard OpenFlow functionality is used and intermediate TCP packets are dropped during handoff transition.
- **Reactive & Proactive (Resume):** Standard OpenFlow protocol is enriched with `Pause` and `Resume` messages. The active TCP flow is paused until the new route was established during handoff.
- **Proactive (Random):** Proactive (Resume) approach is used such that only a randomly chosen half of the routes are established.

The D2D experiment considers three possible handoff scenarios. For a given D2D link, depending on the mobility pattern, single mobility handoff, dual mobility handoff or D2D to D2I handoff may occur. Only the reactive mode is considered for D2D mobility. When single mobility handoff occurs, the resultant set-up is a D2D link

that is controlled by two different eNBs. We assume that this handoff will result in a change of the resource allocation for the D2D data link. The coordinated allocation of the resources for the two eNBs is managed by the RAN controller in this case. When dual mobility is encountered, the control planes of the UEs may both move to the same destination eNB, or to different eNBs. If the move is to the same eNB, only the eNB controllers are involved in the handoff operation. Otherwise, RAN controller is once again involved for coordinated resource allocation. We assume that for half of the cases that result in dual eNB control of the D2D link, a handoff to D2I will be necessary. We assume in this case the RAN and Network controllers are involved in the handoff control along with the eNB controllers.

In order to establish a comparison baseline for the evaluated models, LTE handoff scheme is implemented and evaluated in the conducted experiments. Here it is assumed that the uplink is kept intact and client is served through a low quality link until the handoff transition completes. For the downlink, the flow is dropped and established again, which implies that the LTE uplink follows an identical approach to reactive (Resume) model.

Each experimented model is considered with 3 different variables (rule expire period, handoff period, and user count), downlink and uplink scenarios, and D2I and D2D schemes. The complete set of experimental results are presented and discussed in Tables 1 to 36 and Figures 6 to 41.

Handoff Period	Reactive		Proactive			LTE
	OpenFlow	Resume	OpenFlow	Random	Resume	
1.0	7.98	6.56	4.70	4.13	3.55	6.79
1.5	8.14	6.12	4.84	4.09	3.33	6.15
2.0	8.24	5.89	4.87	4.02	3.18	5.77
3.0	8.43	5.68	4.75	3.83	2.91	5.34
5.0	8.63	5.51	5.18	4.09	2.99	5.12
10.0	8.59	5.23	6.19	4.81	3.43	5.03
15.0	8.30	4.96	7.80	6.06	4.32	5.28
20.0	8.16	4.82	7.36	5.67	3.99	5.01
25.0	8.09	4.74	7.11	5.46	3.81	4.85
30.0	8.01	4.67	6.91	5.29	3.66	4.72

Table 1: The effect of *handoff period* on the average TCP packet *latency* for *D2I uplink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The units are in milliseconds. The numbers are plotted in Figure 6.

Handoff Period	Reactive		Proactive		
	OpenFlow	Resume/LTE	OpenFlow	Random	Resume
1.0	10.15	8.67	5.53	5.22	4.92
1.5	10.38	7.80	5.76	5.13	4.50
2.0	10.52	7.33	5.80	5.01	4.21
3.0	10.78	6.89	5.63	4.71	3.80
5.0	11.06	6.49	6.23	4.99	3.76
10.0	11.01	6.00	7.65	5.85	4.06
15.0	10.61	5.65	9.90	7.41	4.92
20.0	10.41	5.49	9.28	6.91	4.54
25.0	10.30	5.38	8.94	6.63	4.31
30.0	10.20	5.29	8.66	6.40	4.14

Table 2: The effect of *handoff period* on the average TCP packet *latency* for *D2I downlink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The units are in milliseconds. The numbers are plotted in Figure 7.

Rule Expire Period	Reactive		Proactive			LTE
	OpenFlow	Resume	OpenFlow	Random	Resume	
5	8.16	4.82	7.36	5.67	3.99	5.01
10	8.16	4.82	7.36	5.67	3.99	5.01
20	8.16	4.82	5.44	4.12	2.80	4.40
30	8.16	4.82	4.19	3.10	2.01	3.99

Table 3: The effect of *rule expire period* on the average TCP packet *latency* for *D2I uplink* flows. Handoff period and user count are respectively fixed to 20s and 50. The units are in milliseconds. The numbers are plotted in Figure 8.

Rule Expire Period	Reactive		Proactive		
	OpenFlow	Resume/LTE	OpenFlow	Random	Resume
5	10.41	5.49	9.28	6.91	4.54
10	10.41	5.49	9.28	6.91	4.54
20	10.41	5.49	6.60	4.96	3.31
30	10.41	5.49	4.85	3.68	2.50

Table 4: The effect of *rule expire period* on the average TCP packet *latency* for *D2I downlink* flows. Handoff period and user count are respectively fixed to 20s and 50. The units are in milliseconds. The numbers are plotted in Figure 9.

User Count	Reactive		Proactive			LTE
	OpenFlow	Resume	OpenFlow	Random	Resume	
10	5.59	3.43	3.58	2.63	1.69	3.25
50	7.98	6.56	4.70	4.13	3.55	6.79
100	11.19	10.51	6.25	6.07	5.89	11.31

Table 5: The effect of *user count* on the average TCP packet *latency* for *D2I uplink* flows. Rule expire and handoff periods are respectively fixed to 1s and 10s. The units are in milliseconds. The numbers are plotted in Figure 10.

User Count	Reactive		Proactive		
	OpenFlow	Resume/LTE	OpenFlow	Random	Resume
10	6.81	4.26	3.99	3.12	2.25
50	10.15	8.67	5.53	5.22	4.92
100	14.65	14.23	7.73	8.06	8.39

Table 6: The effect of *user count* on the average TCP packet *latency* for *D2I downlink* flows. Rule expire and handoff periods are respectively fixed to 1s and 10s. The units are in milliseconds. The numbers are plotted in Figure 11.

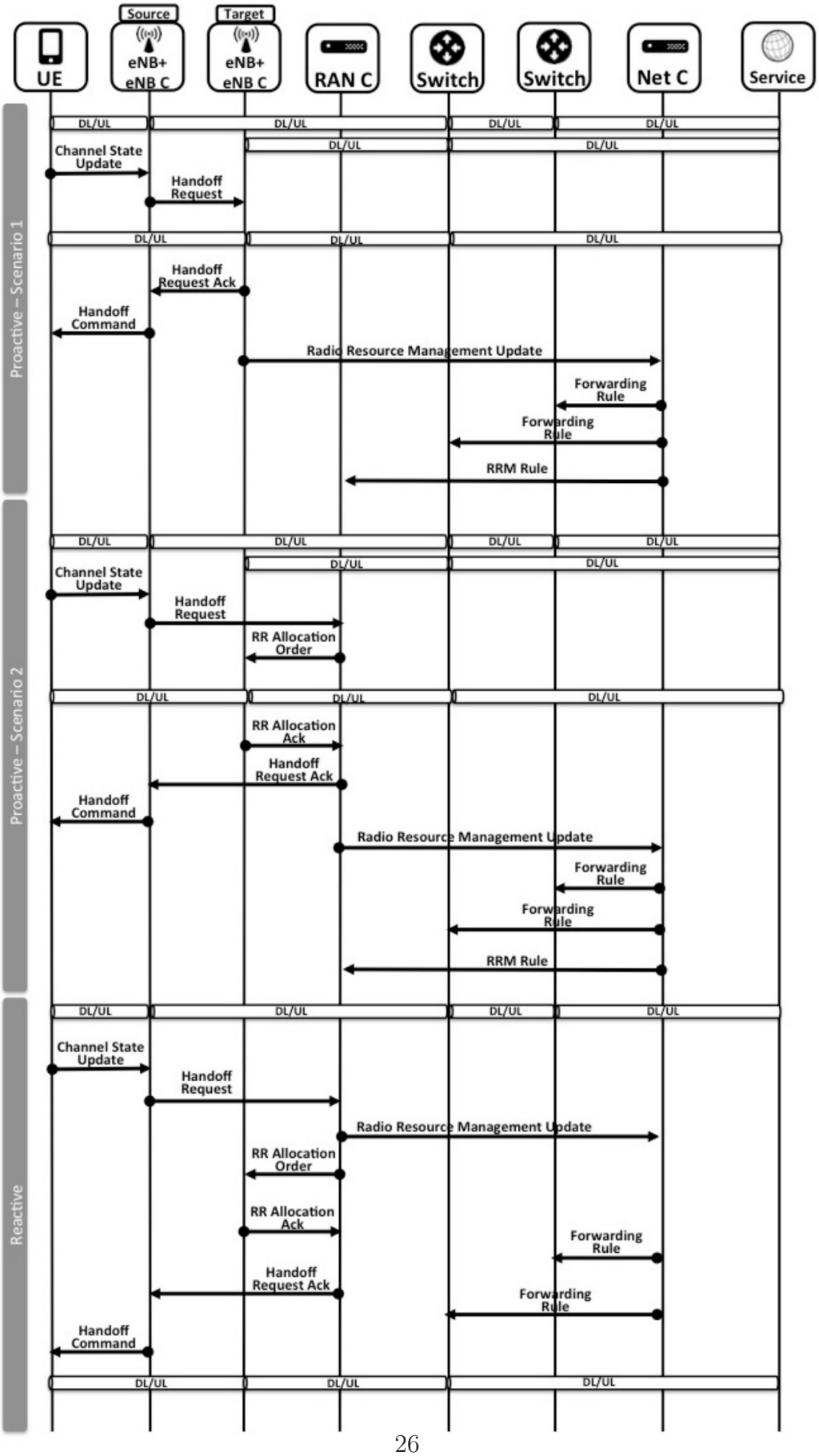


Figure 3: Message flows for proactive and reactive mobility management control of D2I links

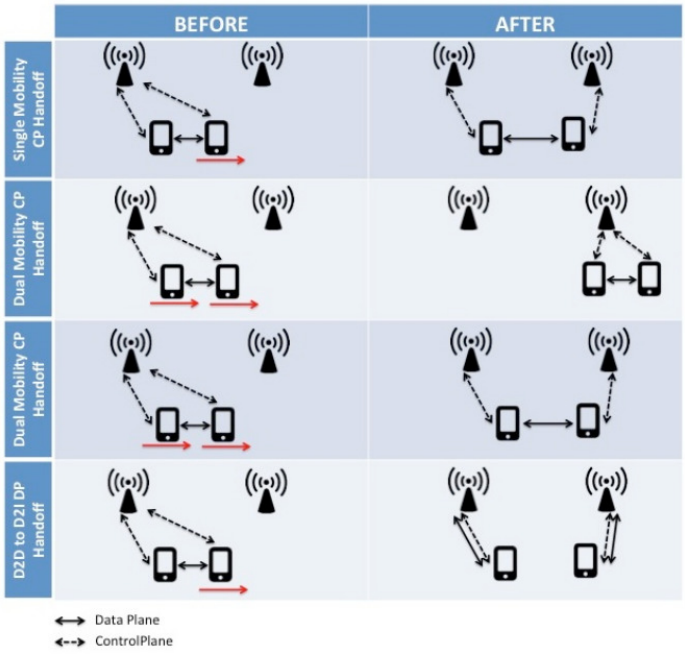


Figure 4: D2D Mobility Scenarios

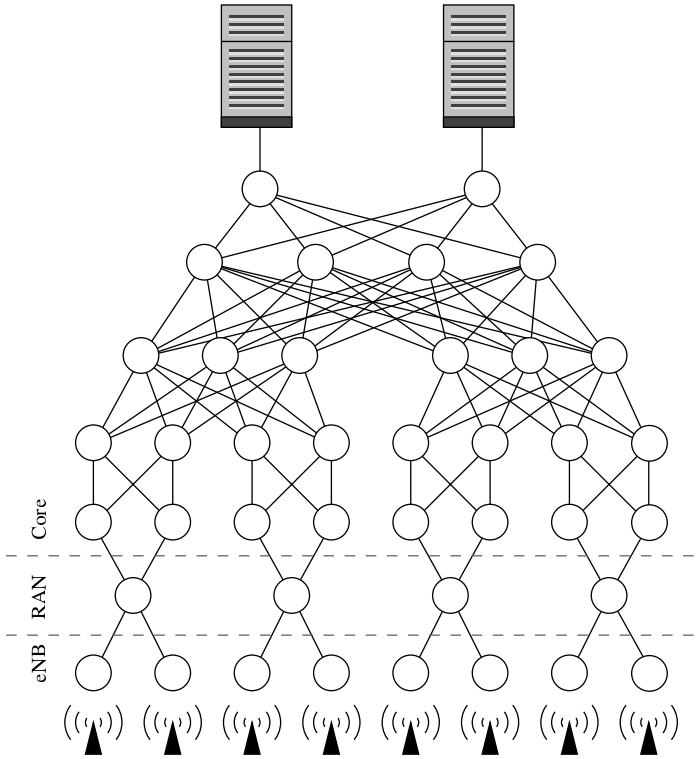


Figure 5: Cellular Network Map for CMaaS Performance Simulations

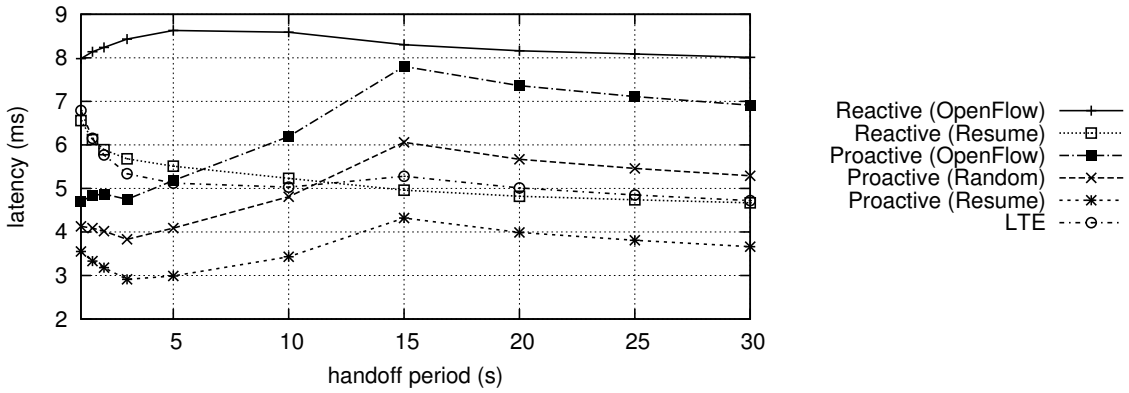


Figure 6: The effect of *handoff period* on the average TCP packet *latency* for *D2I uplink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are tabulated in Table 1.

The latency first decreases due to the less control plane overhead implied by the decrease in the frequency of handoffs. Later on, as the handoff period increases, the probability of a priori computed and established routes to match a flow decreases. This inevitably necessitates a new route establishment and exposes a certain delay. Finally figures stabilize due to the contraction at the control plane overhead. Since Reactive (OpenFlow) mode does not involve a priori route establishment, it is oblivious to these changes and just reacts to the control plane overhead. As expected, LTE approach sits in between reactive and proactive modes. Surprisingly, Reactive (Resume) follows an almost identical path to LTE.

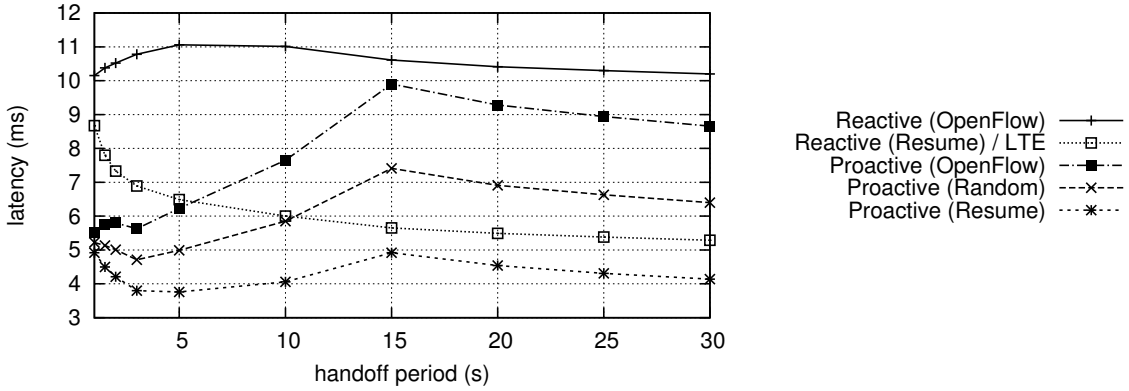


Figure 7: The effect of *handoff period* on the average TCP packet *latency* for *D2I downlink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are tabulated in Table 2.

Numbers follow a similar pattern to their uplink counterpart presented in Figure 6.

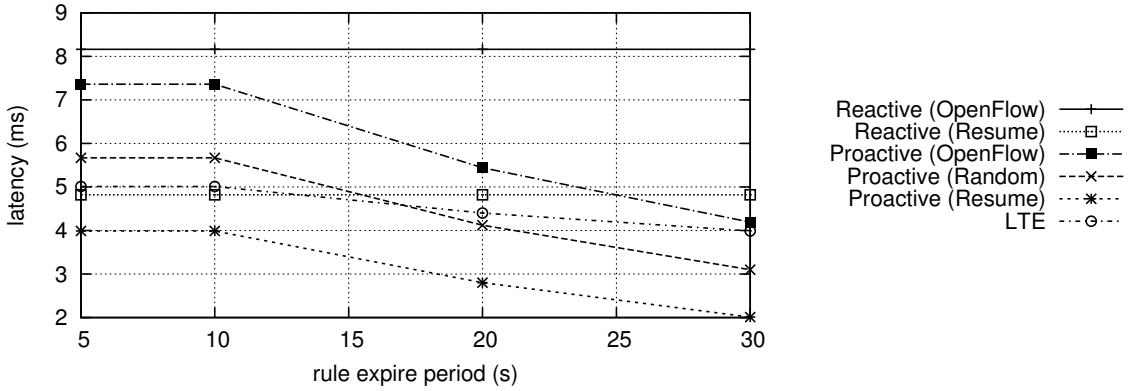


Figure 8: The effect of *rule expire period* on the average TCP packet *latency* for *D2I uplink* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are tabulated in Table 3.

As rule expire period increases, the probability of a priori computed route to survive to match an incoming flow request increases. This in return causes the flow to reuse an already established route and imposes almost zero routing overhead at the control plane. For reactive approaches, changes in the rule expire period do not have an effect on the latency.

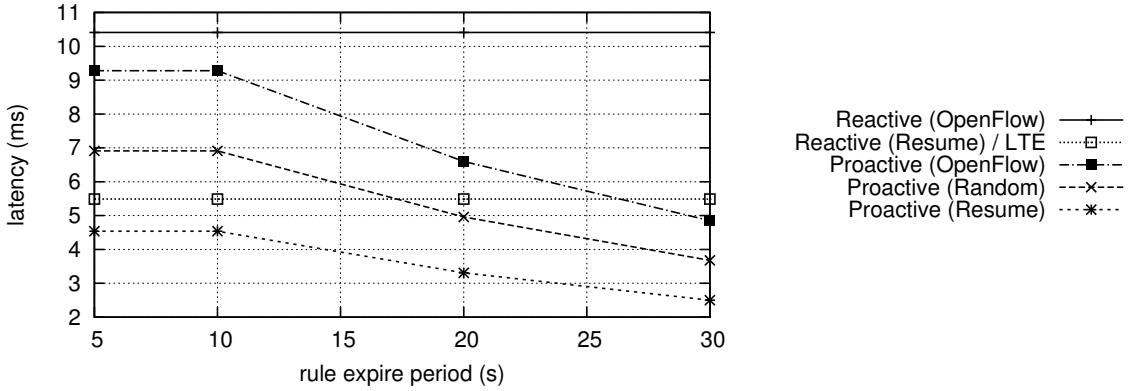


Figure 9: The effect of *rule expire period* on the average TCP packet *latency* for *D2I downlink* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are tabulated in Table 4.

Numbers follow a similar pattern to their uplink counterpart presented in Figure 8.

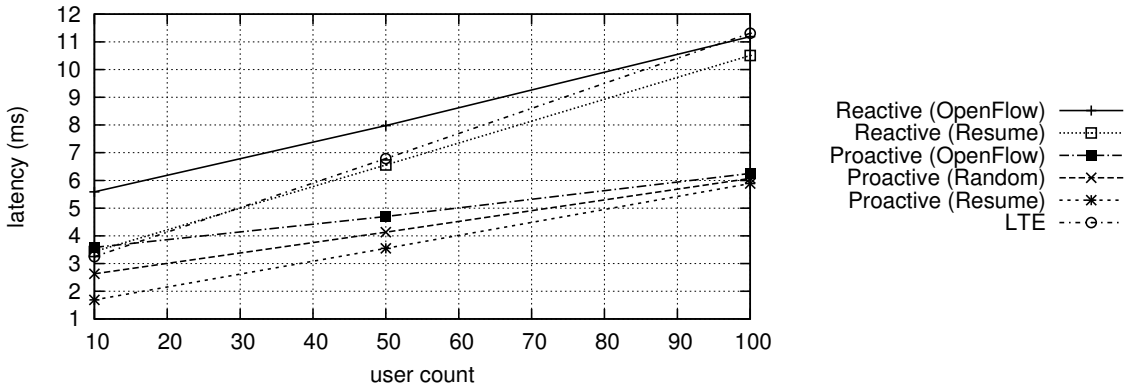


Figure 10: The effect of *user count* on the average TCP packet *latency* for *D2I uplink* flows. Rule expire and handoff periods are respectively fixed to 1s and 10s. The numbers are tabulated in Table 5.

The increase in the user count shows almost a linear increase in the latency due to the implied increase in the control plane overhead. LTE and Reactive (Resume) modes follow almost identical patterns and Proactive modes appear to be less affected by the increase in the number of users.

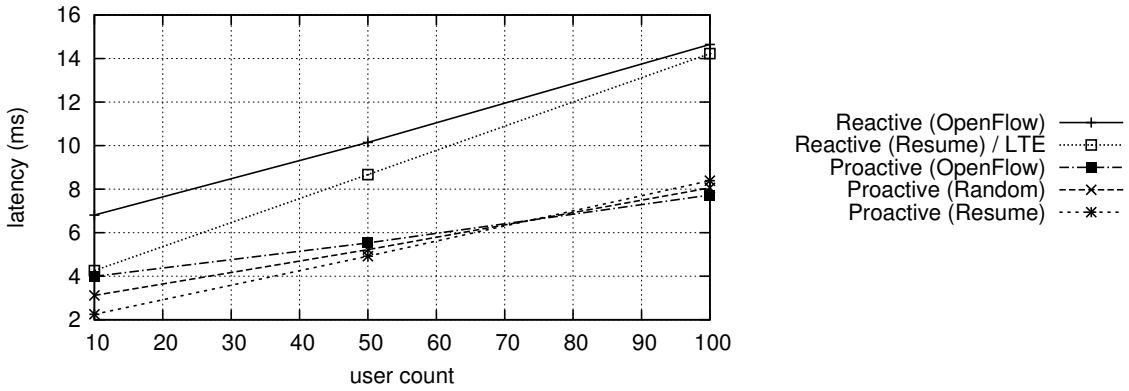


Figure 11: The effect of *user count* on the average TCP packet *latency* for *D2I downlink* flows. Rule expire and handoff periods are respectively fixed to 1s and 10s. The numbers are tabulated in Table 6.

Numbers follow a similar pattern to their uplink counterpart presented in Figure 10. Additionally, figures show that downlink traffic is slightly more sensitive than uplink traffic to an increase in the number of users.

Handoff Period	Reactive		Proactive			LTE
	OpenFlow	Resume	OpenFlow	Random	Resume	
1.0	62.41	62.62	62.91	63.00	63.09	62.59
1.5	62.38	62.69	62.89	63.00	63.12	62.69
2.0	62.36	62.73	62.88	63.01	63.15	62.75
3.0	62.34	62.76	62.90	63.04	63.19	62.81
5.0	62.30	62.79	62.84	63.00	63.17	62.85
10.0	62.31	62.83	62.68	62.89	63.11	62.86
15.0	62.35	62.87	62.43	62.70	62.97	62.82
20.0	62.38	62.89	62.50	62.76	63.02	62.86
25.0	62.39	62.90	62.54	62.79	63.05	62.89
30.0	62.40	62.91	62.57	62.82	63.07	62.91

Table 7: The effect of *handoff period* on the average TCP data *throughput* for *D2I uplink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The units are in megabits per second. The numbers are plotted in Figure 12.

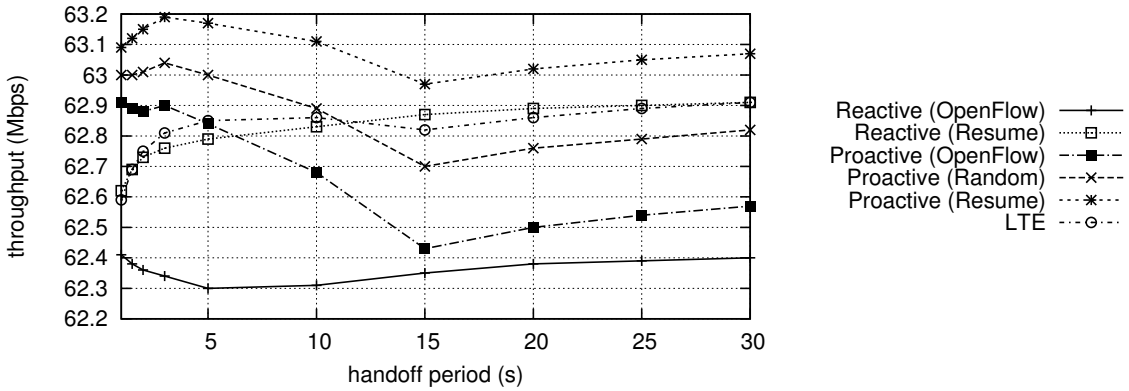


Figure 12: The effect of *handoff period* on the average TCP data *throughput* for *D2I uplink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are tabulated in Table 7.

Plotted numbers follow the reverse pattern of their latency counterparts presented in Figure 1. The numbers first increase due to imposed low overhead at the control plane, then decrease due to a priori rule mismatches, and finally stabilize. As anticipated, Reactive schemes are only affected by the control plane overhead, since they do not perform any a priori route establishment.

Handoff	Reactive		Proactive		
Period	OpenFlow	Resume/LTE	OpenFlow	Random	Resume
1.0	62.07	62.30	62.78	62.83	62.88
1.5	62.04	62.43	62.75	62.84	62.94
2.0	62.01	62.50	62.74	62.86	62.99
3.0	61.97	62.57	62.77	62.91	63.05
5.0	61.93	62.63	62.67	62.86	63.06
10.0	61.94	62.71	62.46	62.73	63.01
15.0	62.00	62.76	62.11	62.49	62.88
20.0	62.03	62.79	62.20	62.57	62.94
25.0	62.05	62.81	62.26	62.61	62.97
30.0	62.06	62.82	62.30	62.65	63.00

Table 8: The effect of *handoff period* on the average TCP data *throughput* for *D2I downlink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The units are in megabits per second. The numbers are plotted in Figure 13.

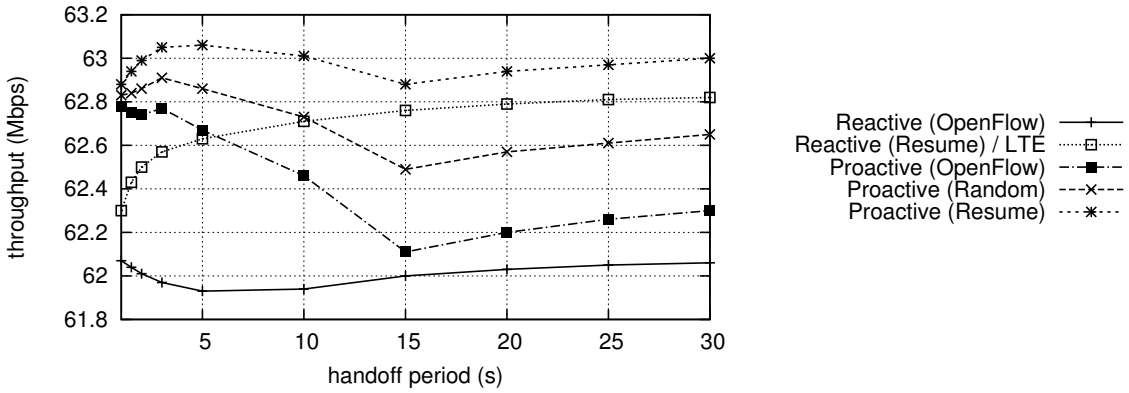


Figure 13: The effect of *handoff period* on the average TCP data *throughput* for *D2I downlink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are tabulated in Table 8.

Numbers follow a similar pattern to their uplink counterpart presented in Figure 12.

Rule Expire	Reactive		Proactive			LTE
Period	OpenFlow	Resume	OpenFlow	Random	Resume	
5	62.38	62.89	62.50	62.76	63.02	62.86
10	62.38	62.89	62.50	62.76	63.02	62.86
20	62.38	62.89	62.80	63.00	63.20	62.96
30	62.38	62.89	62.99	63.16	63.32	63.02

Table 9: The effect of *rule expire period* on the average TCP data *throughput* for *D2I uplink* flows. Handoff period and user count are respectively fixed to 20s and 50. The units are in megabits per second. The numbers are plotted in Figure 14.

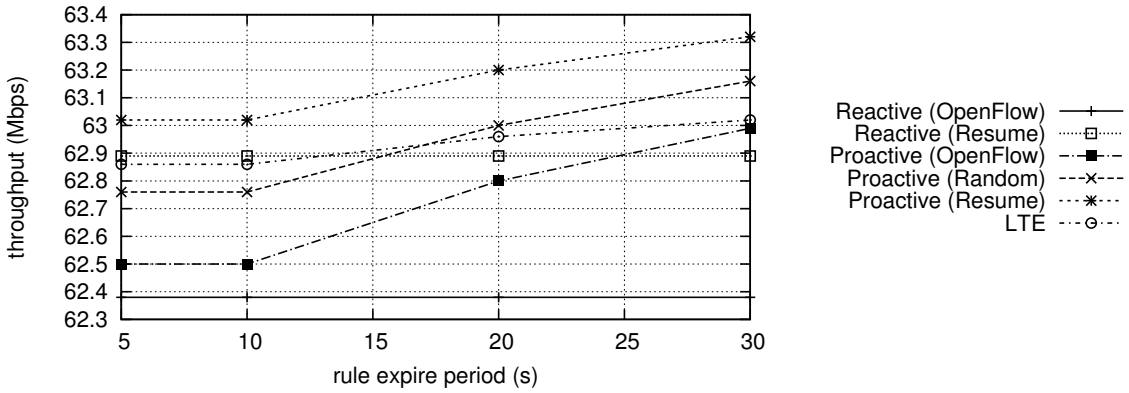


Figure 14: The effect of *rule expire period* on the average TCP data *throughput* for *D2I uplink* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are tabulated in Table 9.

Plotted numbers follow the reverse pattern of their latency counterparts presented in Figure 3. That is, the throughput improves with the increase in the rule expire period due to less rule mismatch at the data plane. Further, as anticipated, Reactive modes are not affected by the rule expire period.

Rule Expire Period	Reactive		Proactive		
	OpenFlow	Resume/LTE	OpenFlow	Random	Resume
5	62.03	62.79	62.20	62.57	62.94
10	62.03	62.79	62.20	62.57	62.94
20	62.03	62.79	62.62	62.87	63.12
30	62.03	62.79	62.89	63.07	63.25

Table 10: The effect of *rule expire period* on the average TCP data *throughput* for *D2I downlink* flows. Handoff period and user count are respectively fixed to 20s and 50. The units are in megabits per second. The numbers are plotted in Figure 15.

User Count	Reactive		Proactive			LTE
	OpenFlow	Resume	OpenFlow	Random	Resume	
10	62.77	63.11	63.08	63.23	63.37	63.13
50	62.41	62.62	62.91	63.00	63.09	62.59
100	61.91	62.02	62.67	62.70	62.73	61.89

Table 11: The effect of *user count* on the average TCP data *throughput* for *D2I uplink* flows. Rule expire and handoff periods are respectively fixed to 1s and 10s. The units are in megabits per second. The numbers are plotted in Figure 16.

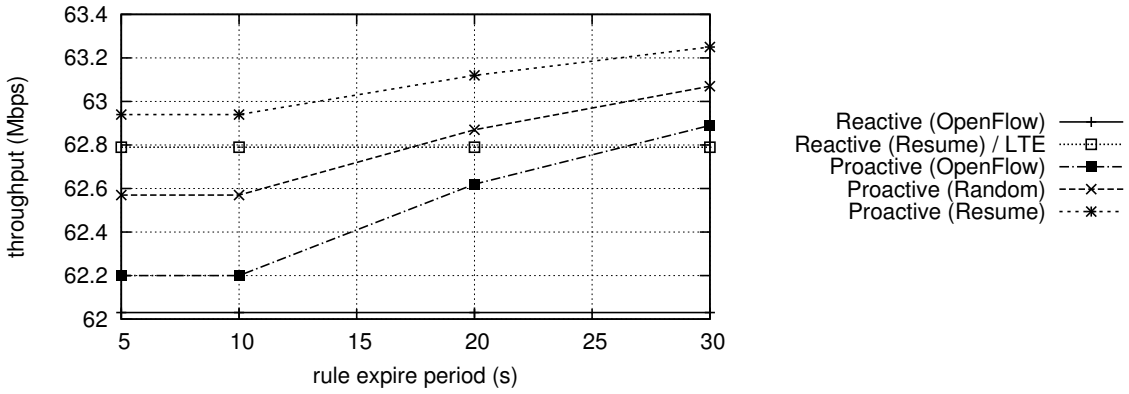


Figure 15: The effect of *rule expire period* on the average TCP data *throughput* for *D2I downlink* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are tabulated in Table 10.

Numbers follow a similar pattern to their uplink counterpart presented in Figure 14.

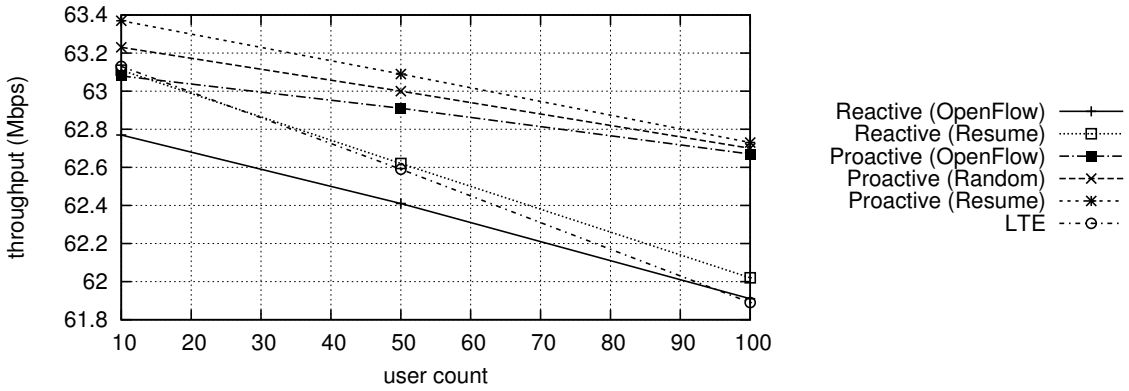


Figure 16: The effect of *user count* on the average TCP data *throughput* for *D2I uplink* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are tabulated in Table 11.

Plotted numbers follow the reverse pattern of their latency counterparts presented in Figure 5. That is, throughput decreases directly propertional to the number of users in the system.

User Count	Reactive		Proactive		
	OpenFlow	Resume/LTE	OpenFlow	Random	Resume
10	62.59	62.98	63.02	63.15	63.29
50	62.07	62.30	62.78	62.83	62.88
100	61.38	61.44	62.44	62.39	62.34

Table 12: The effect of *user count* on the average TCP data *throughput* for *D2I downlink* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The units are in megabits per second. The numbers are plotted in Figure 17.

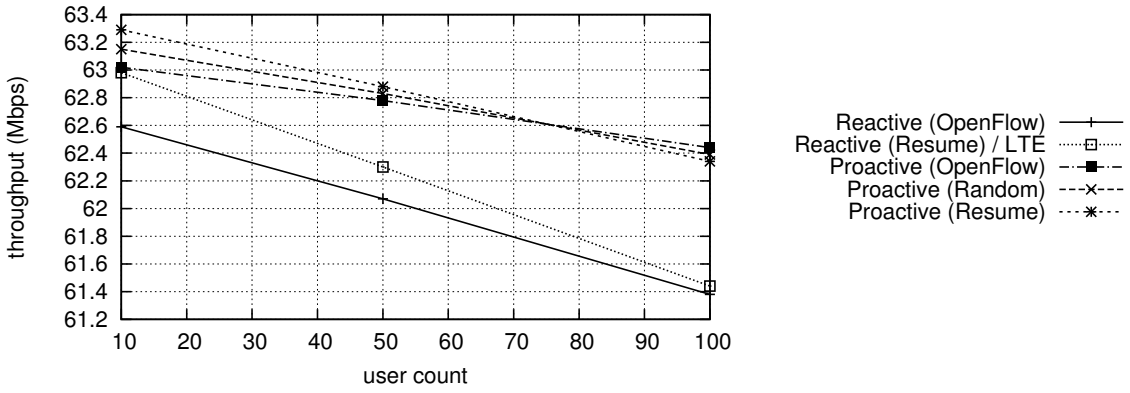


Figure 17: The effect of *user count* on the average TCP data *throughput* for *D2I downlink* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are tabulated in Table 12.

Numbers follow a similar pattern to their uplink counterpart presented in Figure 16.

Handoff	Reactive		Proactive		
Period	OpenFlow	Resume	OpenFlow	Random	Resume
1.0	3,360.52	5,214.81	5,011.79	4,188.31	6,866.08
1.5	2,684.63	3,910.48	4,386.06	3,423.27	5,611.91
2.0	2,285.96	3,197.46	3,984.64	2,986.65	4,896.14
3.0	1,821.31	2,418.76	3,448.36	2,467.94	4,045.81
5.0	1,374.52	1,720.68	2,944.91	2,007.55	3,291.07
10.0	930.66	1,089.85	2,283.00	1,489.74	2,442.19
15.0	751.65	847.74	2,065.07	1,318.31	2,161.16
20.0	662.52	727.24	1,660.82	1,052.58	1,725.54
25.0	612.86	660.07	1,435.49	904.45	1,482.70
30.0	574.00	607.60	1,260.04	789.12	1,293.64

Table 13: The effect of *handoff period* on the average *control plane load* for *D2I uplink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The units are in processed messages per second. The numbers are plotted in Figure 18.

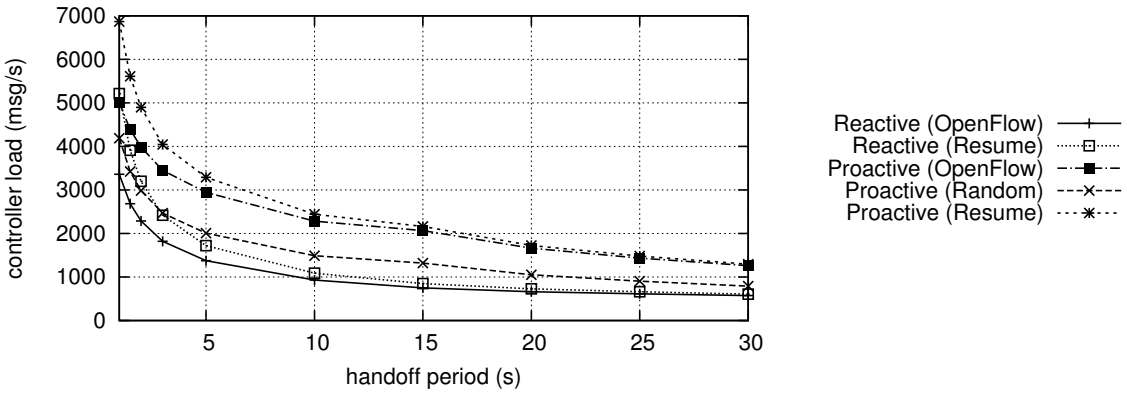


Figure 18: The effect of *handoff period* on the average *control plane load* for *D2I uplink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are tabulated in Table 13.

The increase in handoff period has a certain positive effect on the control plane load due to the lower number of requests imposed on the system. Nevertheless, after a certain threshold, rule expire period starts to show its effect and triggers more route establishments. That is, rule expire periods fixed to 10s start to fall short of routing handoffs with longer periods. This effect is also observable on the decrease in the control plane load improvement.

Handoff	Reactive		Proactive		
Period	OpenFlow	Resume	OpenFlow	Random	Resume
1.0	3,361.82	6,143.25	4,916.07	4,695.48	7,697.50
1.5	2,686.05	4,524.82	4,387.48	3,798.01	6,226.25
2.0	2,287.38	3,654.63	3,986.05	3,265.51	5,353.30
3.0	1,817.75	2,718.59	3,444.80	2,650.84	4,345.64
5.0	1,372.45	1,894.40	2,942.83	2,113.52	3,464.78
10.0	931.45	1,169.62	2,283.80	1,538.40	2,521.97
15.0	752.15	895.92	2,065.57	1,347.70	2,209.34
20.0	662.37	759.70	1,660.67	1,072.38	1,758.00
25.0	613.11	683.75	1,435.73	918.89	1,506.37
30.0	574.16	624.43	1,260.20	799.39	1,310.47

Table 14: The effect of *handoff period* on the average *control plane load* for *D2I downlink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The units are in processed messages per second. The numbers are plotted in Figure 19.

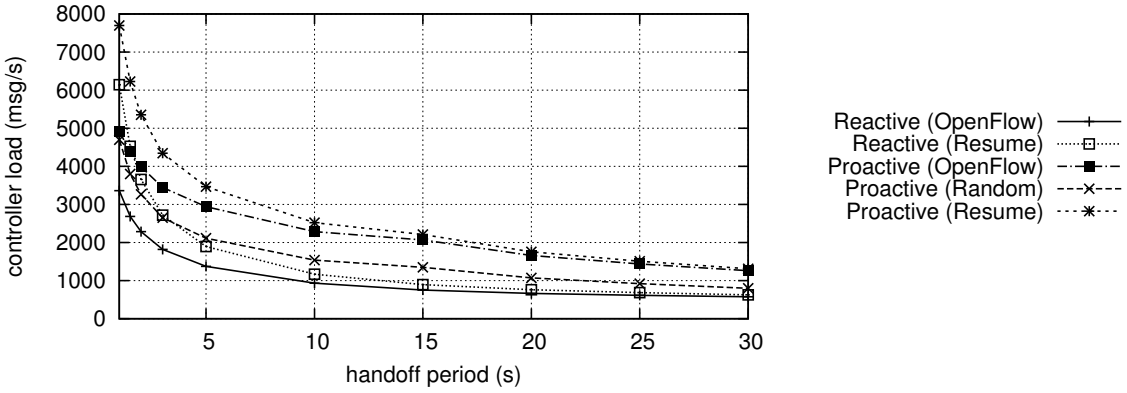


Figure 19: The effect of *handoff period* on the average *control plane load* for *D2I downlink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are tabulated in Table 14.

Numbers follow a similar pattern to their uplink counterpart presented in Figure 18.

Rule Expire Period	Reactive		Proactive		
	OpenFlow	Resume	OpenFlow	Random	Resume
5	662.52	727.24	1,660.82	1052.58	1,725.54
10	662.52	727.24	1,660.82	1052.58	1,725.54
20	662.52	727.24	1,385.24	884.48	1,449.96
30	662.52	727.24	1,219.94	783.64	1,284.66

Table 15: The effect of *rule expire period* on the average *control plane load* for *D2I uplink* flows. Handoff period and user count are respectively fixed to 20s and 50. The units are in processed messages per second. The numbers are plotted in Figure 20.

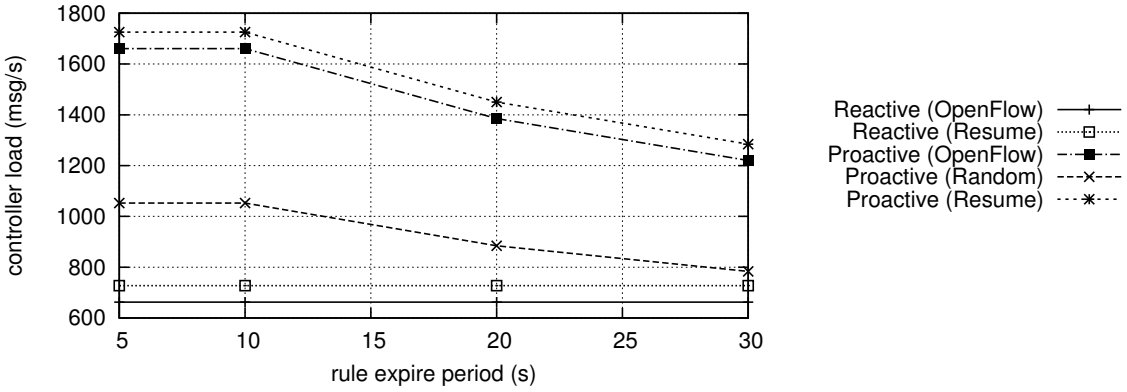


Figure 20: The effect of *rule expire period* on the average *control plane load* for *D2I uplink* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are tabulated in Table 15.

The increase in rule expire period results in rules to persist a longer period of time on the data plane. This in return increases the chance of a flow to match to an existing rule and does not necessitate an extra route establishment in the control plane.

Rule Expire Period	Reactive		Proactive		
	OpenFlow	Resume	OpenFlow	Random	Resume
5	662.37	759.70	1,660.67	1,052.58	1,758.00
10	662.37	759.70	1,660.67	1,052.58	1,758.00
20	662.37	759.70	1,385.07	884.48	1,482.40
30	662.37	759.70	1,219.77	783.64	1,317.10

Table 16: The effect of *rule expire period* on the average *control plane load* for *D2I downlink* flows. Handoff period and user count are respectively fixed to 20s and 50. The units are in processed messages per second. The numbers are plotted in Figure 21.

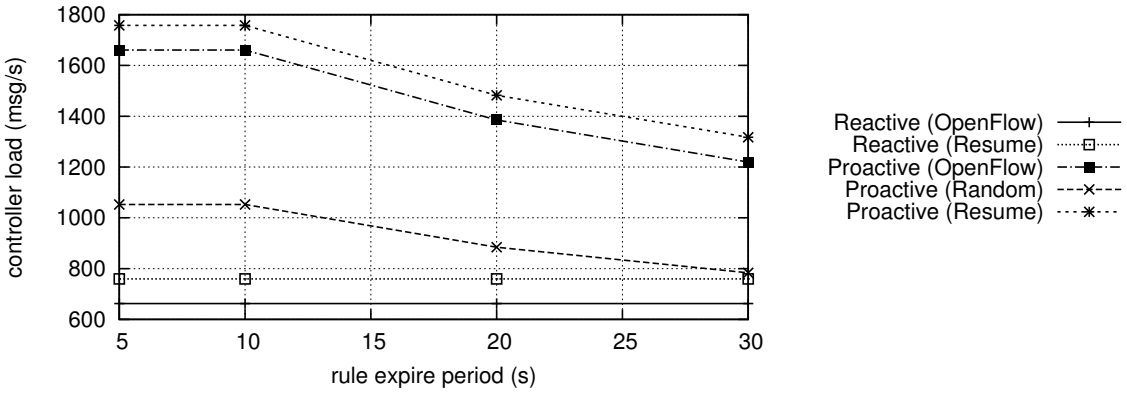


Figure 21: The effect of *rule expire period* on the average *control plane load* for *D2I downlink* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are tabulated in Table 16.

Numbers follow a similar pattern to their uplink counterpart presented in Figure 20.

User Count	Reactive		Proactive		
	OpenFlow	Resume	OpenFlow	Random	Resume
10	669.27	1,038.35	999.02	834.54	1,368.10
50	3,360.52	5,214.81	5,011.79	4,188.31	6,866.08
100	6,750.74	10,479.76	10,073.63	8,419.62	13,802.65

Table 17: The effect of *user count* on the average *control plane load* for *D2I uplink* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The units are in processed messages per second. The numbers are plotted in Figure 22.

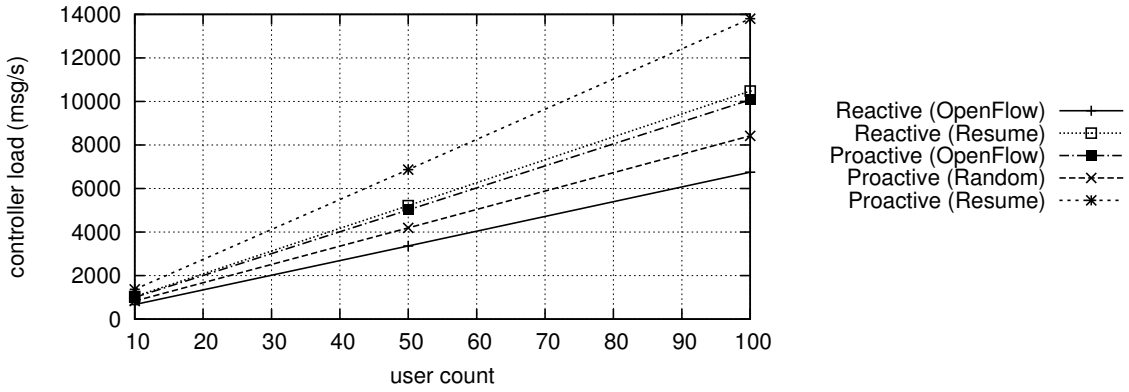


Figure 22: The effect of *user count* on the average *control plane load* for *D2I uplink* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are tabulated in Table 17.

The increase in user count has an anticipated negative effect on the control plane load. That being said, the growth seems to be linear even for Proactive schemes due to the hierarchical expansion of the control plane.

User Count	Reactive		Proactive		
	OpenFlow	Resume	OpenFlow	Random	Resume
10	669.45	1,223.07	999.20	947.22	1,552.82
50	3,361.82	6,143.25	4,916.07	4,695.48	7,697.50
100	6,752.09	12,345.63	10,074.98	9,557.80	15,668.52

Table 18: The effect of *user count* on the average *control plane load* for *D2I downlink* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The units are in processed messages per second. The numbers are plotted in Figure 23.

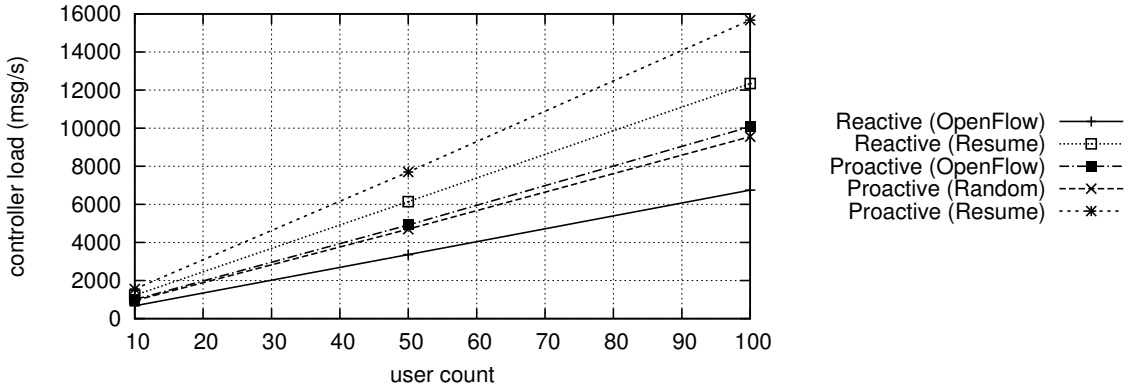


Figure 23: The effect of *user count* on the average *control plane load* for *D2I downlink* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are tabulated in Table 17.

Numbers follow a similar pattern to their uplink counterpart presented in Figure 22.

Handoff	Reactive		Proactive		
Period	OpenFlow	Resume	OpenFlow	Random	Resume
1.0	609.35	945.58	955.92	851.23	1,309.59
1.5	543.44	791.59	938.93	780.88	1,201.35
2.0	494.31	691.41	909.91	726.74	1,118.06
3.0	425.48	565.05	845.08	644.47	991.49
5.0	342.17	428.34	776.05	563.73	867.27
10.0	243.30	284.92	641.09	445.76	685.79
15.0	201.61	227.39	598.82	407.34	626.68
20.0	182.98	200.86	483.72	326.67	502.57
25.0	173.74	187.13	419.96	281.95	433.77
30.0	167.23	177.02	370.37	247.16	380.25

Table 19: The effect of *handoff period* on the average *flow table size* for *D2I uplink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are plotted in Figure 24.

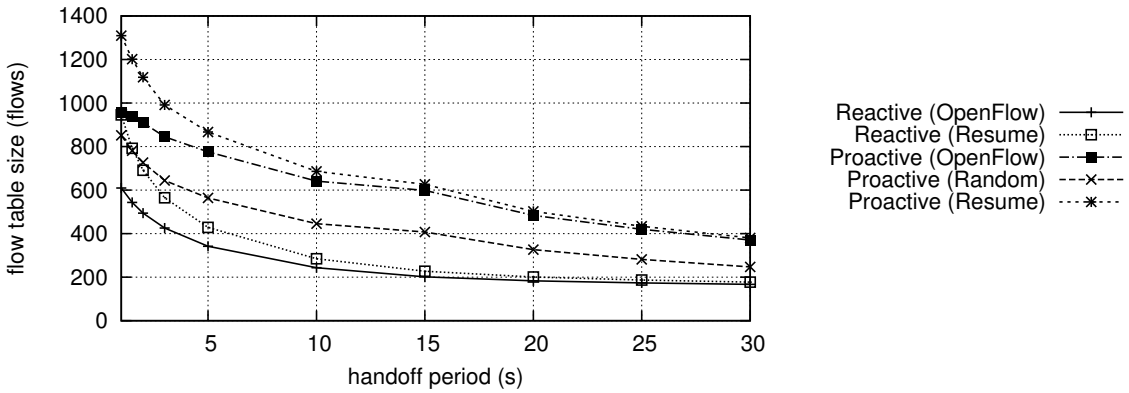


Figure 24: The effect of *handoff period* on the average *flow table size* for *D2I uplink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are tabulated in Table 19.

These figures show almost an identical pattern to their control plane load counterparts in Figure 18. Note that Reactive schemes require less number of active rules in the flow table since they do not employ any redundant route establishments to reduce latency and throughput.

Handoff Period	Reactive		Proactive		
	OpenFlow	Resume	OpenFlow	Random	Resume
1.0	517.46	945.58	836.38	851.23	1,309.59
1.5	469.91	791.59	846.56	780.88	1,201.35
2.0	432.74	691.41	832.50	726.74	1,118.06
3.0	377.81	565.05	785.96	644.47	991.49
5.0	310.32	428.34	736.62	563.73	867.27
10.0	226.90	284.92	621.03	445.76	685.79
15.0	190.90	227.39	585.90	407.34	626.68
20.0	175.13	200.86	474.75	326.67	502.57
25.0	167.80	187.13	413.43	281.95	433.77
30.0	162.77	177.02	365.66	247.16	380.25

Table 20: The effect of *handoff period* on the average *flow table size* for *D2I downlink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are plotted in Figure 25.

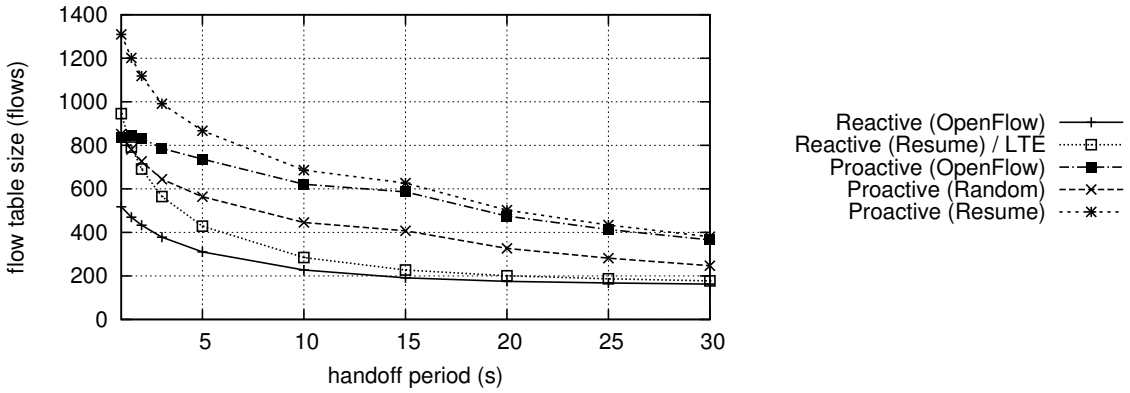


Figure 25: The effect of *handoff period* on the average *flow table size* for *D2I downlink* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are tabulated in Table 20.

Numbers follow a similar pattern to their uplink counterpart presented in Figure 24.

Rule Expire Period	Reactive		Proactive		
	OpenFlow	Resume	OpenFlow	Random	Resume
5	91.74	100.70	242.16	163.54	251.60
10	182.98	200.86	483.72	326.67	502.57
20	365.19	400.86	791.67	538.63	828.66
30	546.99	600.42	1,028.10	703.72	1,082.64

Table 21: The effect of *rule expire period* on the average *flow table size* for *D2I uplink* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are plotted in Figure 26.

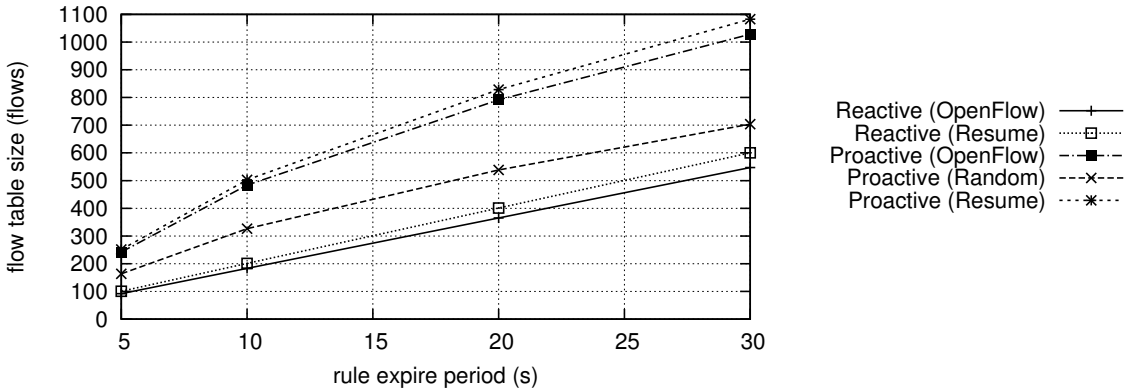


Figure 26: The effect of *rule expire period* on the average *flow table size* for *D2I uplink* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are tabulated in Table 21.

As anticipated the increase in rule expire period implies a longer persistence period for the rules, hence results in larger flow table sizes.

Rule Expire Period	Reactive		Proactive		
	OpenFlow	Resume	OpenFlow	Random	Resume
5	87.80	100.70	237.67	163.54	251.60
10	175.13	200.86	474.75	326.67	502.57
20	349.50	400.86	774.25	538.63	828.66
30	523.50	600.42	1,002.64	703.72	1,082.64

Table 22: The effect of *rule expire period* on the average *flow table size* for *D2I downlink* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are plotted in Figure 27.

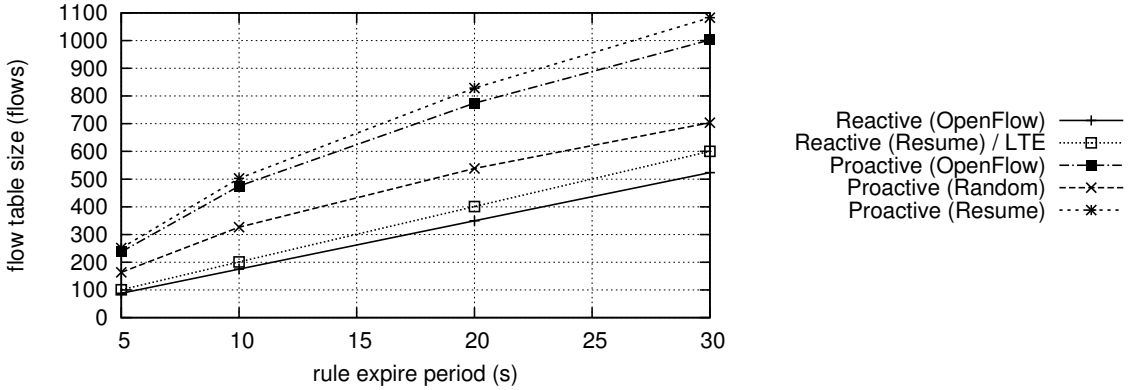


Figure 27: The effect of *rule expire period* on the average *flow table size* for *D2I downlink* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are tabulated in Table 22.

Numbers follow a similar pattern to their uplink counterpart presented in Figure 24.

User Count	Reactive		Proactive		
	OpenFlow	Resume	OpenFlow	Random	Resume
10	121.89	189.11	191.34	170.32	262.03
50	609.35	945.58	955.92	851.23	1,309.59
100	1,223.66	1,899.60	1,920.93	1,710.81	2,632.02

Table 23: The effect of *user count* on the average *flow table size* for *D2I uplink* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are plotted in Figure 28.

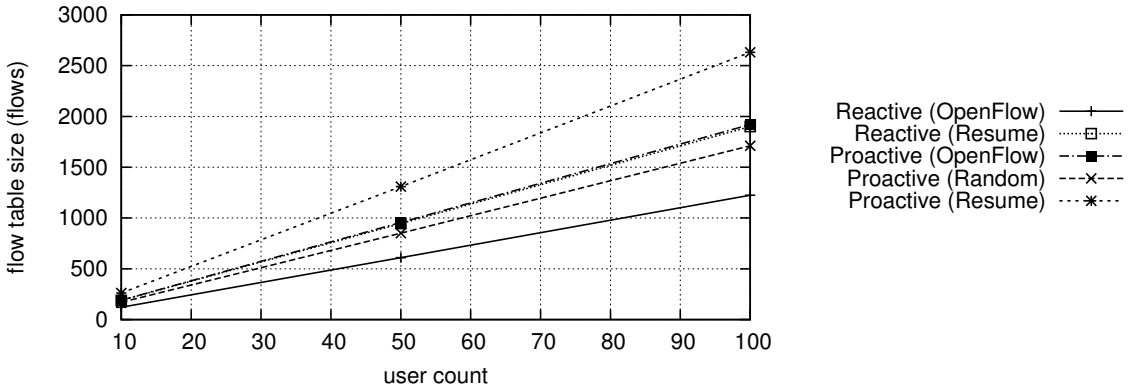


Figure 28: The effect of *user count* on the average *flow table size* for *D2I uplink* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are tabulated in Table 23.

The increase in the number of users processed by the network incurs a set of extra flows for each new user. Hence, as anticipated the flow table size increases accordingly.

User Count	Reactive		Proactive		
	OpenFlow	Resume	OpenFlow	Random	Resume
10	103.51	189.11	168.61	170.32	262.03
50	517.46	945.58	836.38	851.23	1,309.59
100	1,038.93	1,899.60	1,692.41	1,710.81	2,632.02

Table 24: The effect of *user count* on the average *flow table size* for *D2I downlink* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are plotted in Figure 29.

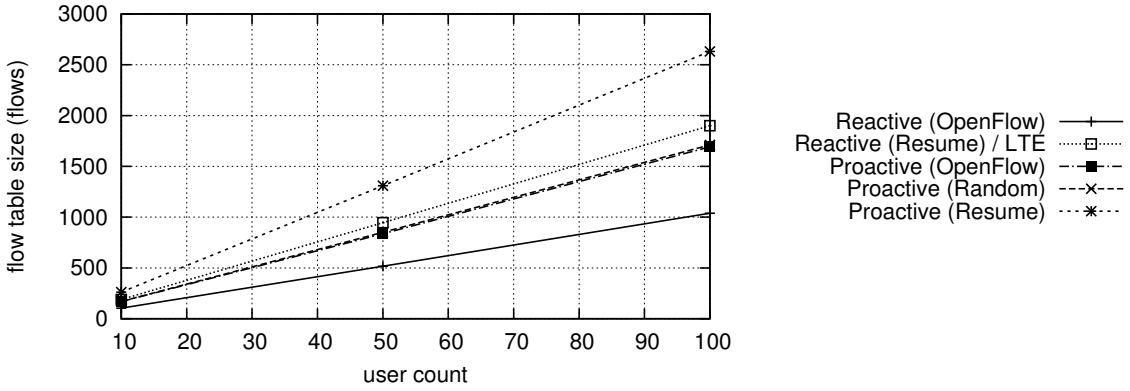


Figure 29: The effect of *user count* on the average *flow table size* for *D2I downlink* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are tabulated in Table 24.

Numbers follow a similar pattern to their uplink counterpart presented in Figure 28.

Handoff Period	OpenFlow	Resume
1.0	9.40	5.07
1.5	9.74	5.27
2.0	10.06	5.46
3.0	10.18	5.57
5.0	10.65	5.88
10.0	10.87	6.09
15.0	11.11	6.27
20.0	10.54	5.97
25.0	9.91	5.60
30.0	10.01	5.68

Table 25: The effect of *handoff period* on the average TCP packet *latency* for *D2D* flows. Rule expire period and user count are respectively fixed to 10s and 50. The units are in milliseconds. The numbers are plotted in Figure 30.

Rule Expire Period	OpenFlow	Resume
5	10.54	5.97
10	10.54	5.97
20	9.38	5.27
30	8.49	4.73

Table 26: The effect of *rule expire period* on the average TCP packet *latency* for *D2D* flows. Handoff period and user count are respectively fixed to 20s and 50. The units are in milliseconds. The numbers are plotted in Figure 31.

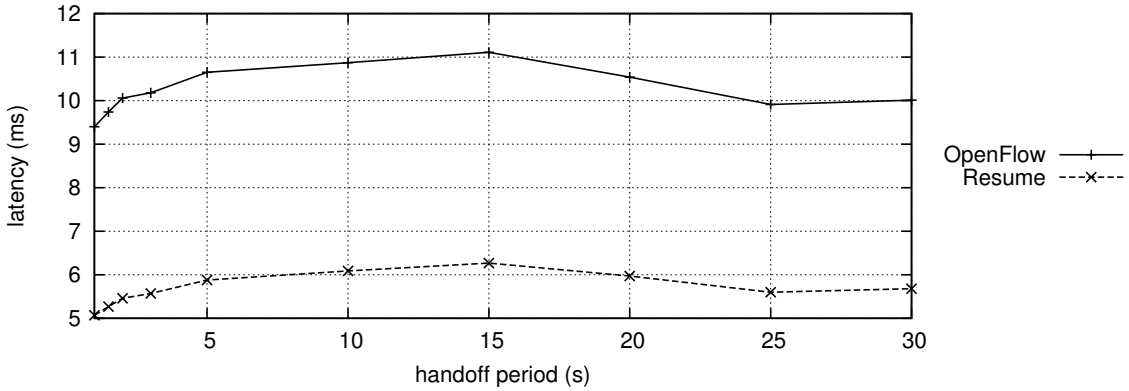


Figure 30: The effect of *handoff period* on the average TCP packet *latency* for *D2D* flows. Rule expire period and user count are respectively fixed to 10s and 50. The units are in milliseconds. The numbers are tabulated in Table 25.

D2D flows experience a certain improvement as the handoff period increases and accordingly control plane load decreases. That being said, after a certain threshold, this improvement retracts backwards due to the flow mismatches caused by expired rules. Note that pausing flows during handoff transition performs sigfinicantly better than dropping the TCP packets.

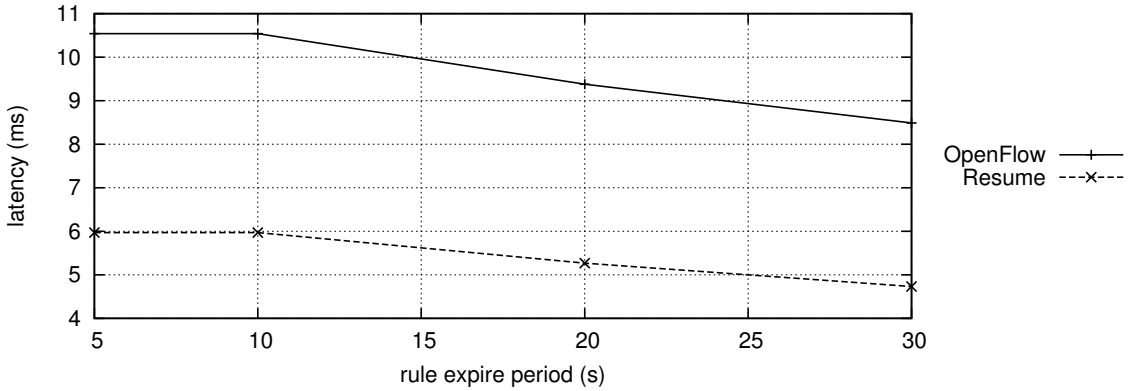


Figure 31: The effect of *rule expire period* on the average TCP packet *latency* for *D2D* flows. Handoff period and user count are respectively fixed to 20s and 50. The units are in milliseconds. The numbers are tabulated in Table 26.

Figures point out that the increase in rule expire period has a positive effect on the TCP packet latency. That is, longer persistence periods of rules imply higher reusage of these rules, which in return reduces latency.

User Count	OpenFlow	Resume
10	5.84	3.11
50	9.40	5.07
100	14.29	7.55

Table 27: The effect of *user count* on the average TCP packet *latency* for *D2D* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The units are in milliseconds. The numbers are plotted in Figure 32.

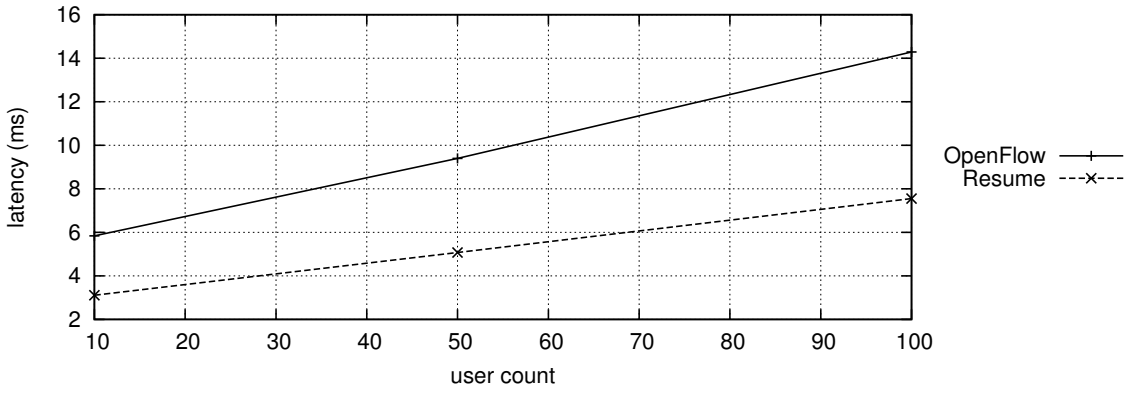


Figure 32: The effect of *user count* on the average TCP packet *latency* for *D2D* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The units are in milliseconds. The numbers are tabulated in Table 27.

The increase in number of users creates a certain load on the control plane and this returns back as an impact to the D2D flow latency.

Handoff Period	OpenFlow	Resume
1.0	62.19	62.85
1.5	62.13	62.82
2.0	62.08	62.79
3.0	62.07	62.78
5.0	61.99	62.73
10.0	61.96	62.70
15.0	61.92	62.67
20.0	62.01	62.72
25.0	62.11	62.77
30.0	62.09	62.76

Table 28: The effect of *handoff period* on the average TCP packet *throughput* for *D2D* flows. Rule expire period and user count are respectively fixed to 10s and 50. The units are in megabits per second. The numbers are plotted in Figure 33.

Rule Expire Period	OpenFlow	Resume
5	62.01	62.72
10	62.01	62.72
20	62.19	62.82
30	62.33	62.91

Table 29: The effect of *rule expire period* on the average TCP packet *throughput* for *D2D* flows. Handoff period and user count are respectively fixed to 20s and 50. The units are in megabits per second. The numbers are plotted in Figure 34.

User Count	OpenFlow	Resume
10	62.73	63.16
50	62.19	62.85
100	61.43	62.47

Table 30: The effect of *user count* on the average TCP packet *throughput* for *D2D* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The units are in megabits per second. The numbers are plotted in Figure 35.

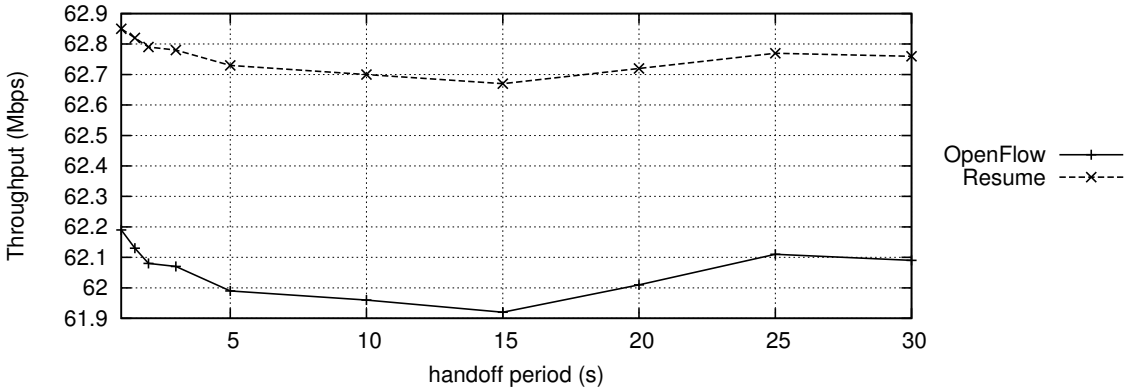


Figure 33: The effect of *handoff period* on the average TCP packet *throughput* for *D2D* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are tabulated in Table 28.

The increase in the handoff frequency also increases the probability of a device pair to establish a D2D communication. Later on, the lower the frequency is, the longer the devices will stay in D2I mode and suffer from low quality controller response times. But after some threshold, D2I performance will also improve due to less control plane load and the figures will again return back to their initial values.

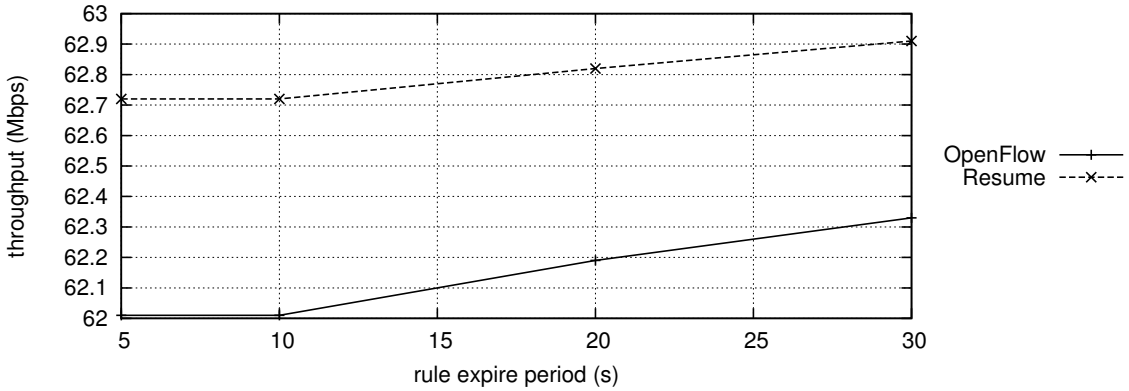


Figure 34: The effect of *rule expire period* on the average TCP packet *throughput* for *D2D* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are tabulated in Table 29.

Making rules persist longer on the data plane by increasing their expiration periods will make them used more than once for the very same flow after every route establishment triggered by a handoff.

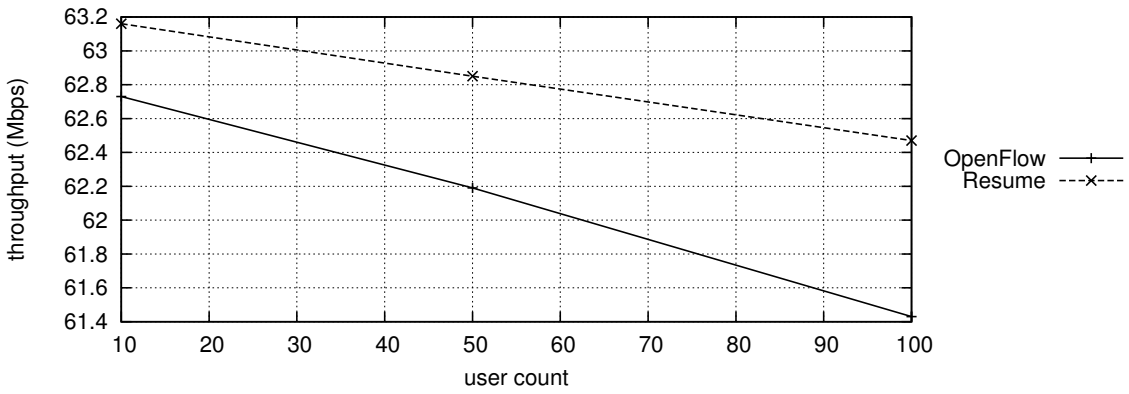


Figure 35: The effect of *user count* on the average TCP packet *throughput* for *D2D* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are tabulated in Table 30.

The increase in the user count has a negative impact on the TCP flow throughput. Nevertheless, paused flows apperantly are less affected by this impact.

Handoff Period	OpenFlow	Resume
1.0	273.28	287.27
1.5	225.05	234.37
2.0	195.43	202.42
3.0	154.47	159.13
5.0	117.34	120.14
10.0	73.87	75.27
15.0	56.50	57.43
20.0	42.39	43.09
25.0	33.29	33.81
30.0	28.34	28.80

Table 31: The effect of *handoff period* on the average *control plane load* for *D2D* flows. Rule expire period and user count are respectively fixed to 10s and 50. The units are in megabits per second. The numbers are plotted in Figure 36.

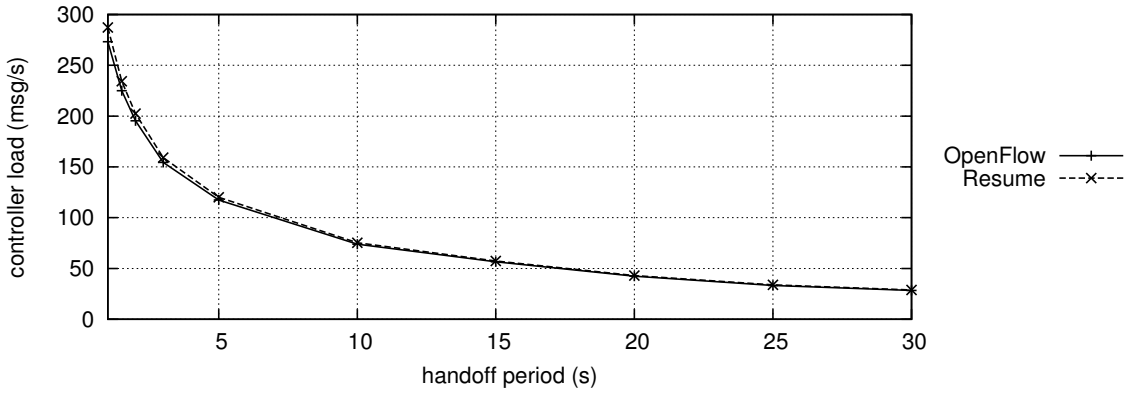


Figure 36: The effect of *handoff period* on the average *control plane load* for *D2D* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are tabulated in Table 31.

As anticipated, the increase in handoff period result in less load on the control plane. Further, numbers show that the enhanced OpenFlow protocol with `Pause` and `Resume` messages incur almost no extra cost compared to its standard OpenFlow counterpart.

Rule Expire Period	OpenFlow	Resume
5	42.39	43.09
10	42.39	43.09
20	36.85	37.55
30	32.48	33.18

Table 32: The effect of *rule expire period* on the average *control plane load* for *D2D* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are plotted in Figure 37.

User Count	OpenFlow	Resume
10	54.60	57.40
50	273.28	287.27
100	545.80	573.77

Table 33: The effect of *user count* on the average *control plane load* for *D2D* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are plotted in Figure 38.

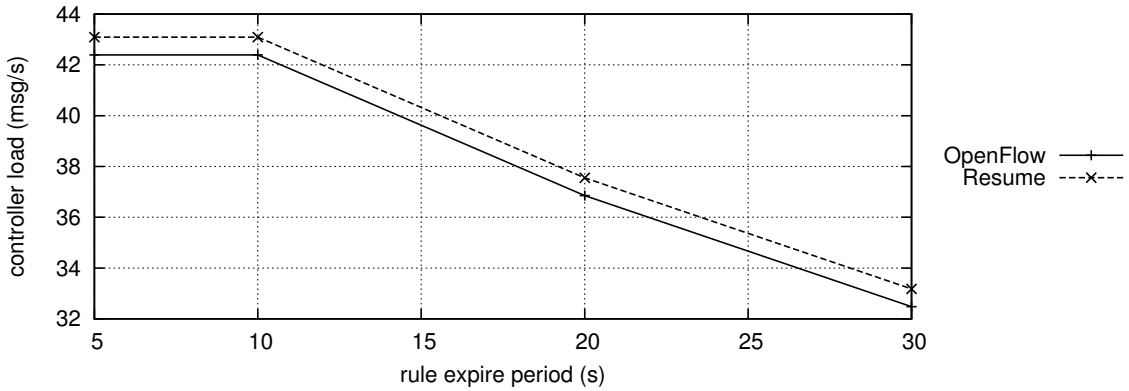


Figure 37: The effect of *rule expire period* on the average *control plane load* for *D2D* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are tabulated in Table 32.

As rules persist in longer periods, the more they will be used by new flows, and eventually the less the control plane load will be triggered.

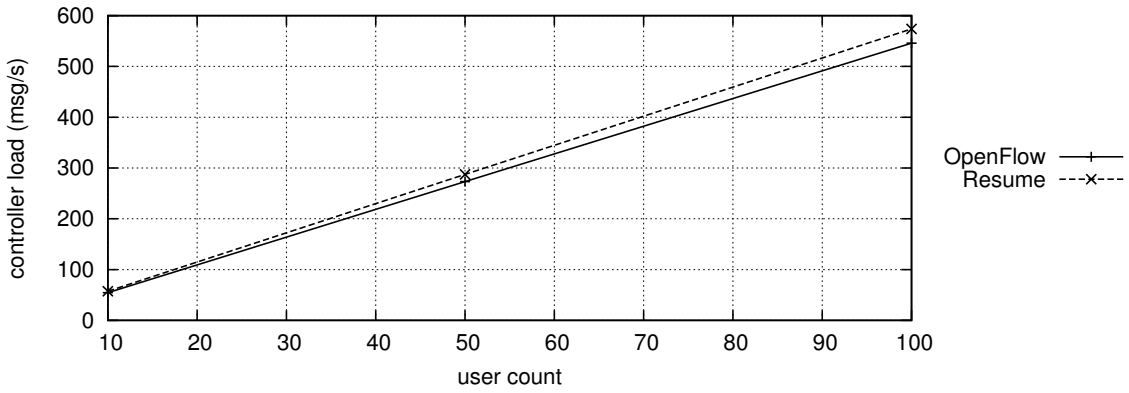


Figure 38: The effect of *user count* on the average *control plane load* for *D2D* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are tabulated in Table 33.

The number of users present in the system is directly proportional with the load imposed on the control plane.

Handoff Period	OpenFlow	Resume
1.0	65.88	69.25
1.5	58.71	61.14
2.0	53.28	55.19
3.0	44.37	45.71
5.0	35.42	36.26
10.0	23.41	23.85
15.0	18.32	18.62
20.0	14.00	14.23
25.0	11.06	11.23
30.0	9.56	9.72

Table 34: The effect of *handoff period* on the average *flow table size* for *D2D* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are plotted in Figure 39.

Rule Expire Period	OpenFlow	Resume
5	7.31	7.43
10	14.00	14.23
20	23.55	24.00
30	30.55	31.21

Table 35: The effect of *rule expire period* on the average *flow table size* for *D2D* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are plotted in Figure 40.

For a D2I link, the current LTE handoff operation is a make-before-break scheme. On the other hand, the reactive CMaaS is a break-before-make scheme, whereas the proactive CMaaS is a make-before-handoff request scheme. As such, the current LTE operation sits somewhat in between the reactive and proactive modes. Specifically in LTE, once the source eNB decides to make a handoff, it continues to serve the UE until the end-to-end route for the UE via the target UE is established. However, during the handoff operation, this link will be of low quality. Using the same network topology and UE mobility pattern, we conduct experiments to assess the LTE handoff performance. We assume an average 100 ms handoff operation duration [14], during which the link via the new eNB is to be established. During this time, we assume a

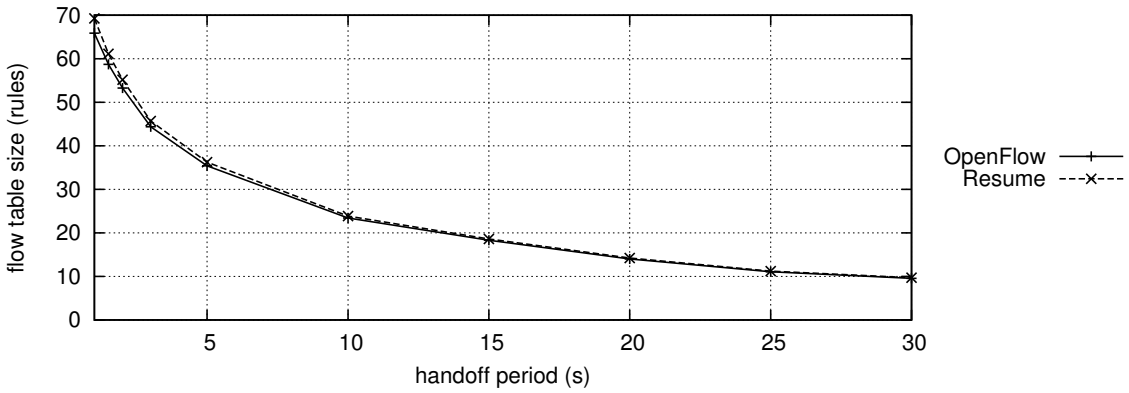


Figure 39: The effect of *handoff period* on the average *flow table size* for *D2D* flows. Rule expire period and user count are respectively fixed to 10s and 50. The numbers are tabulated in Table 34.

The decrease in the mobility of clients results in lower number of flows in the network.

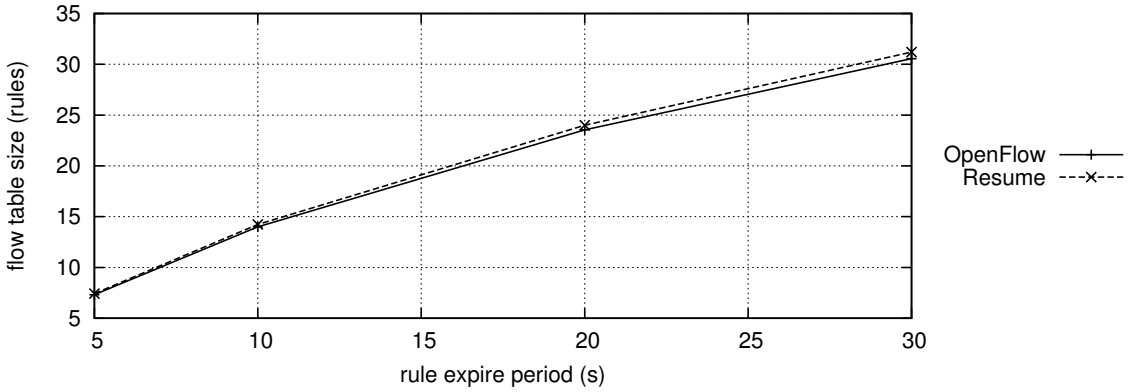


Figure 40: The effect of *rule expire period* on the average *flow table size* for *D2D* flows. Handoff period and user count are respectively fixed to 20s and 50. The numbers are tabulated in Table 35.

The persistence period of the rules increases the presence of a rule in the data plane. As a result of this, the average number of rules in the data plane increases.

User Count	OpenFlow	Resume
10	1.85	1.88
50	7.31	7.43
100	14.22	14.45

Table 36: The effect of *user count* on the average *flow table size* for *D2D* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are plotted in Figure 41.

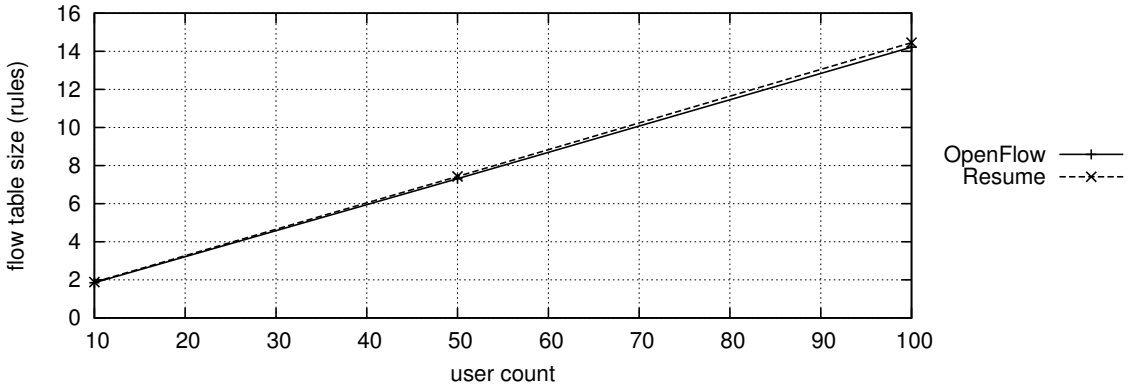


Figure 41: The effect of *user count* on the average *flow table size* for *D2D* flows. Handoff and rule expire periods are respectively fixed to 1s and 10s. The numbers are tabulated in Table 36.

As the number of users in the network increases, more rules are pushed to the data plane.

40% drop in the wireless channel capacity between the source eNB and the UE [4]. This is incorporated into the experiment by assuming that the wireless link allows only 60% of the full-quality link TCP throughput during the 100 ms of the handoff operation.

For D2I handoffs, we observe that the proactive CMaaS achieves RTT delays that are surpassing the performances of the LTE handoff procedure as well as the reactive CMaaS. The reactive CMaaS on the other hand, experiences larger RTT delays, increasing with higher controller hierarchy. The proactive CMaaS observes minimal loss in TCP throughput due to handoffs, while the corresponding loss is higher in the reactive CMaaS. The superior data plane performance of the proactive CMaaS comes at the expense of an increased complexity control plane as evidenced by the larger number of control plane rules that need to be computed and communicated by the controller every second compared to the reactive CMaaS. As expected, the current LTE scheme provides a performance that lies in between those of the proactive and reactive CMaaS. The programmability nature of the proposed architecture allows for either of the three (or other) schemes to be deployed where this decision may be

made on a per flow basis. It should be noted here that today's SDN controllers are capable of responding up to 1,000,000 flows per second when run on Amazon's Elastic Computer Cloud using a Cluster Compute Eight Extra Large instance, containing 16 physical cores from 2 x Intel Xeon E5-2670 processors, 60.5GB of RAM, using a 64-bit Ubuntu 11.10 VM image [15]. Despite larger incurred delays, the reactive CMaaS is still useful for an MSP for deployment for delay-tolerant services especially when control plane capacity is critically needed for some other application in the network or when some of the eNBs are critically loaded.

For D2D handoffs, we observe that the RTT delays are very small. Indeed, the delay is only 3.1 ms when there are 10 eNBs for each RAN controller. We observe that the TCP flow throughputs of the mobile D2D links are comparable to those of the mobile D2I links.

The `Pause` and `Resume` functionality introduced to the standard OpenFlow protocol performed superior compared to other approaches evaluated. Additionally, it causes an almost negligible impact on the data plane load. Randomly picking half of the routes proposed by Proactive (Resume) approach also reduces the data plane overhead and results in promising performance sitting in between Proactive (OpenFlow) and Proactive (Resume) approaches.

CHAPTER IV

FLOW-LEVEL PERFORMANCE IN MULTI-DOMAIN SOFTWARE-DEFINED NETWORKS

The centralized control plane constitutes the fundamental assertion of the software-defined networks, which manifest the decoupling of the network orchestration off from the forwarding elements. That being said, there are both technical (e.g., scalability, reliability) and economical (e.g., separation of administrative domains) concerns inherent to networks that necessitate a distributed deployment. In this chapter, we investigate the effect of multi-domain control planes on the flow-level network performance. For this purpose, we first evaluate the problem of partitioning a network into multiple domains and present an integer-linear program solution. Next, we share our emulation results produced using a custom distributed control plane implementation and investigate the effect of network partitioning on flow setup time, latency, and throughput performance using a multitude of partitions, network topologies, routing algorithms, and inter-domain network state broadcast periods. Results point out that partitioning provides notable improvements, and after a certain number of domains, the return starts to diminish. Contrary to the conventional belief, flow setup time experiences a significant improvement and the throughput essentially stays constant.

4.1 Introduction

Network forwarding elements (e.g., switches, routers, wireless access points) are still shipped with the entire routing logic built in. This model, in addition to being vulnerable to vendor lock in, hampers the innovation at the network layer. The *software-defined networking* paradigm, which is introduced to address these concerns, manifests

the handling of the packet forwarding and the routing logic at seperate layers called *data and control planes*, respectively. In this scheme, data plane equipments are purposed with the task of forwarding the packets and the control plane is put in charge of orchestrating (e.g., populating the forwarding tables, collecting the port statistics) the data plane through a standardized interface. Recent decade witnessed numerous million dollar success stories (e.g., [16, 17]) emanated from this paradigm shift.

In a typical setting, the control plane is put in charge of a single administrative domain and realized by a cluster of *controller* servers hosting a multitude of control applications (e.g., routing, quality of service, filtering). The size of the cluster can vary from a single machine to a rack of dedicated servers depending on the anticipated scalability and reliability measures at the control plane. This scheme presents a perfect match for local area networks such that the forwarding equipment is set to reside on a single control domain. In fact, a diverse set of studies in the literature (e.g., [18, 19, 20, 21]) investigated the performance of single-domain control planes composed of distributed controllers and it motivated many successful deployments (e.g.,[22, 17]) within the industry.

The single-domain networks carry out a good job at providing a central orchestration portal for the local area network installations like data center or campus networks. That being said, recent developments in the internet service provider and mobile operator core networks [17, 23, 24] that consider the adaptation of software-defined networking open up a new series of questions that necessitate multi-domain setups. That is, the control plane is anticipated to be capable of handling stringent flow setup time and latency requirements of a data plane spread over a wide geographical region. Further, partial exposure of the network internals and the sharing frequency of this information between domains adds an extra layer of complexity to the problem.

Motivated from the aforementioned concerns, the main contribution of this chapter is to investigate the effect of multi-domain software-defined networks on flow-level performance and present empirical results emulated using a custom distributed controller implementation. For this purpose, we first analyze the problem of partitioning a network into multiple domains, investigate corner cases, and present an integer-linear program solution. Next, we introduce the multi-domain network architecture implementation employed as our test bed. Finally, we present our emulation results generated using a comprehensive set of partitions, network topologies, routing engines and inter-domain network state broadcast periods.

This chapter is organized as follows. In Section 4.2, we present an outline of the investigated architecture and discuss its implications on certain routing metrics. Next, in Section 4.3, we describe the implementation details of the test bed and used routing algorithms. In Section 4.4, we present and discuss the data set and experimental results. Finally, in Sections 4.5 and 5, we provide an extensive literature survey and conclude the chapter, respectively.

4.2 Architecture

In this section, we detail the modelled software-defined networking architecture purposed to evaluate the flow-level performance of a set of partition configurations with different routing schemes and link state update periods. We first investigate the properties of the data and control planes, next focus on the partitioning of the network topology into disjoint domains, and finally discuss the effect of partially shared network state information and switch statistics polling frequency on routing decisions.

A representative model of the employed emulation architecture is given in Figure 42. Here a network composed of 20 switches (i.e., $s_1, s_2, \ldots, s_{20}$) is partitioned into three disjoint domains, where each domain is orchestrated by a particular controller (i.e., c_1, c_2, c_3). Edge-switches (i.e., $s_5, s_8, s_9, s_{13}, s_{14}$) are denoted in dashed

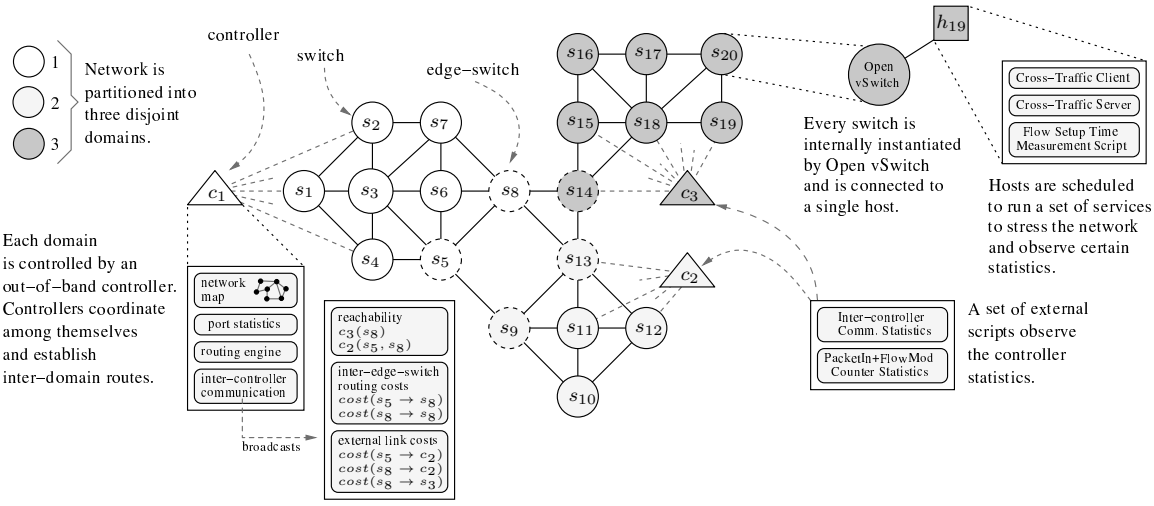


Figure 42: A sample representation of the employed emulation architecture. Here a network composed of 20 switches is partitioned into three disjoint domains, where each domain is orchestrated by a particular controller. A multitude of scripts are employed to observe the system statistics.

circles. We purpose a multitude of scripts both in the network and outside the network to observe the network and controller statistics.

4.2.1 Control & Data Planes

In the proposed model, we envision an architecture, where the data plane is constituted by the switches geographically spread over a region. The switches are grouped into the *domains* such that each domain corresponds to a certain set of IPv4 subnets in the network. The switches in each domain are directly connected to the controller responsible for the orchestration of the corresponding domain. Further, forwarding tables of a switch are populated by the controller responsible of the domain the switch is connected. While there exists multiple protocols (e.g., BGP, OpenFlow, Cisco OnePK) for interfacing controllers with switches, without loss of generality, we employ OpenFlow protocol for programming the switches.

In each domain, controllers periodically poll the port traffic statistics of the switches and take routing decisions based on these collected statistics. Note that the directly collected statistics of the in-domain switches are sufficient for performing

intra-domain routing. That being said, in order to orchestrate the routing decisions spread over multiple domains connected to external controllers, a controller would also need to have a certain amount of information on the network state of the external domains and the adjacency relationship between these domains. For this purpose, in our model, we envision that controllers periodically *broadcast* (1) the domains (i.e., network subnets) connected to them, (2) inter-edge-switch routing costs within domains, (3) external domains adjacent to them, and (4) costs of the links connected to the external domains. Note that the employed broadcast messages encapsulate the partial exposure of the domain internals to its neighbours approach, which resembles the de facto internet connectivity protocol BGP. Controllers receiving external broadcast messages build the global domain connectivity map and enhance it incrementally by the partial network state information (i.e., inter-edge-switch routes) of external domains carried in the broadcast messages.

Figure 42 depicts that c_1 broadcasts its reachability (i.e., s_5 is connected to c_2 and s_8 is connected to both c_2 and c_3), inter-edge-switch routing costs (i.e., cost of routing a packet from s_5 to s_8 and vice versa), and external link costs (i.e., cost of forwarding a packet over an external link). Say a packet arrives to c_2 and is destined to an external domain that is reachable by both c_1 and c_3. Here, c_2 takes into account the most recent broadcasts transmitted by c_1 and c_3 to decide whether it should route the packet through c_1 or c_3. Consequently, c_2 routes the packet to an in-domain edge-switch and hands the packet off to the next domain along the route to the packet destination.

Physical connectivity of the controllers in a distributed control plane can be established by means of either *in-band* (i.e., controllers use the underlying data plane for inter-controller communication) or *out-of-band* communication (i.e., controllers are connected through a separate network). Both design choices have its advantages and disadvantages and found numerous applications in a multitude of deployments. We

believe this discussion is out of the scope of this work and deserves an in-depth study. That being said, in our model, without loss of generality, the control plane employs out-of-band communication, which is known to be a common practice in networks composed of multiple domains [17].

In a network at this scale, resilience of the control plane to failures is anticipated to be an essential piece of the whole puzzle. An inaccessible or malfunctioning controller can yield catastrophic damages ranging from putting the domain into an inoperative state to injecting malicious broadcast messages triggering external controllers to make unexpected and faulty routing decisions, which eventually can lead to the collapse of the entire network. This facet of the distributed control planes was addressed in our prior work [21]. Here, it has been shown that a multitude of controllers can be purposed to orchestrate a single domain with necessary scalability and reliability measures such that the switches are allowed to dynamically migrate from one controller to another instantaneously and the control plane capacity can be on-the-fly upgraded or downgraded without interrupting the data plane.

4.2.2 Network Partitioning

Switches spread over a geographical region (i.e., WAN) introduces certain latency requirements, which are not present in local-area networks. Particularly, internet service provider and mobile operator core networks necessitate the partitioning of the network due to – in addition to management – stringent latency and scability requirements. In [25], authors investigate the effect of a set of controller placements algorithms on the data plane to control plane path latency. In our model, we evaluate the flow-level performance of a multitude of partitioning configurations with various metrics using similar partitioning algorithms given in [25].

In graph theoratical terms, we *partition* the network topology into contiguous and mutually exhaustive *part*s, where each part corresponds to a certain control *domain*.

Throughout this dissertation, we used the words “part” and “domain” interchangeably.

4.2.3 Routing

In our setup, controllers have complete control over the domains connected to them, including the capability of populating the switch forwarding tables, realizing the switch connectivity map and collecting the switch port statistics. Controllers take in-domain routing decisions based on the collected traffic statistics from the switches. In order to route flows across external domains – that is, domains connected to the external controllers – controllers rely on the partial network state information revealed by the received broadcast messages. In this scheme, both the polling frequency of the switch port statistics in each domain and the broadcast messages of the controllers add an extra layer of ambiguity to the global network state information populated by a controller, which has a direct impact on the network performance.

The size of the domains in the network plays a significant role for exposing the internals of the domain as well. That is, considering that a domain exposes only its inter-edge-switch routes in the broadcast messages, the revealed information contains more details as the domains shrink. Hence, the number of domains in the network adds another layer of ambiguity to the global network state information of the controllers.

In order the have a comprehensive observation on the aforementioned actors on routing, we evaluate the performance metrics for various polling and broadcast frequency, and partition size configurations in the conducted tests.

4.3 Experimental Setup

In this section, we detail the experimental setup including the software and hardware configurations. Next, we present the used network partitioning and routing algorithms and their implementations. Finally, we discuss the employed cross-traffic generation

framework including the used performance metrics.

4.3.1 Software & Hardware Configuration

We employ the de facto SDN prototyping framework Mininet 2.1.0 [26] for emulating the data plane in the conducted tests. Internally, Mininet purposes OS-level network virtualization features to isolate the network namespaces of the virtual hosts and interface the switches with each other. Consequently, the rest of the network communication mechanics for OSI Layer 3 and 4 gets provided by the operating system defaults. This feature of Mininet greatly simplifies the deployment of real-world software on the emulation test bed and allows to perform practical measurements.

We configured Mininet to use Open vSwitch 2.0.0 [27] for the switching of the network packets. Internally, Mininet commands Open vSwitch to multiplex the switch data planes on a single Open vSwitch process instance. That being said, Open vSwitch 2.0.0 comes with multi-threading support by default and hence exploits the available multi-core processing resources up to its maximum capacity.

IBM ILOG CPLEX version 12.1.0 is used with default settings to solve the k-median problem for partitioning the network topologies.

Tests are conducted on a single IBM System x3650 M4 with 6 physical Intel Xeon 2.4 GHz processors. The system is set to run Ubuntu Linux (13.10) with Oracle Java Virtual Machine 1.7.0-17. All deployed software components are configured to exploit the underlying 64bit architecture, if possible.

4.3.2 Control Plane Implementation

We purposed a customized fork of the Java-based Floodlight controller [28] to implement an out-of-band distributed control plane. First, controllers are initially provided the list of domains (e.g., subnets such as 9.1.0.0/24) that they are supposed to orchestrate. Next, they employ IP multicasting via IGMPv3 [29] to interface with each other. Here broadcast messages are represented by a Java class attached with the

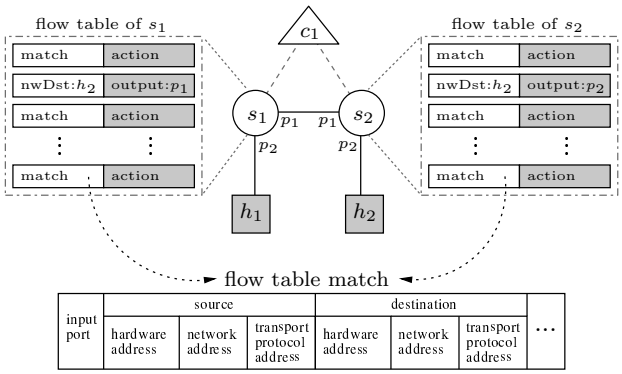


Figure 43: An OpenFlow configuration composed of two switches (s_1, s_2), two hosts (h_1, h_2) and a single controller (c_1). Flows originating from h_1 are directed to h_2 through rules populated at the switch flow tables.

information of (1) the connected domains, (2) the inter-edge-switch routing costs on these domains, (3) the adjacent external domains, and (4) the forwarding costs of the external links. Controllers periodically serialize this class and broadcast it over the control plane. Further, controllers exploit LLDP messages attached with the information of the domain identitities to determine the inter-domain edge switches.

4.3.3 South-Bound Protocol

In this study, controllers are configured to employ the OpenFlow [30] protocol 1.0 to command the data plane. In OpenFlow, flows are represented by a custom tuple and switches are provided a flow table composed of flow-action pairs. Rules in the flow tables are configured to expire either by an idle or hard timeout, whichever comes first. Upon receiving a packet, a switch first tries to associate the packet with a match in its flow table and applies the action pointed by the match. If the switch fails to associate the packet with a match, it forwards the packet to the controller in a *PacketIn* message. Upon receiving the PacketIn, controller computes an appropriate route for the flow and pushes necessary flow table updates (i.e., *FlowMod*s) to the corresponding switches.

In order to construct the in-domain network map and extract the traffic statistics, controllers periodically poll the switches. Here, standard LLDP messages reveal the

inter-switch connectivity in the network. In addition, controllers employ the sampled port transmission rates averaged over the used update period to determine the bandwidth usage of links.

A sample OpenFlow network configuration composed of two switches (s_1, s_2) and a central controller (c_1) is depicted in Figure 43. Here the switch flow tables are populated such that the flows destined to h_2 are first directed to s_2 through p_1 of s_1 and then finally forwared to p_2 of s_2.

4.3.4 Network Partitioning Algorithm

In our experimental setup, we employ *average-case latency metric* [25] to partition the network into disjoint domains. Here, for a given network graph $G(V, E)$, where the edge weights represent propagation latencies and $d(v, s)$ denotes the shortest path from node $v \in V$ to $s \in V$, the *average propagation latency* for a placement $S \subseteq V$ of controllers is given as follows:

$$L_{avg}(S) = \frac{1}{|V|} \sum_{v \in V} \min_{s \in S} d(v, s)$$

In the *average-case latency metric*, the goal is to find a placement S such that $|S| = k$ and $L_{avg}(S)$ is minimum, where k is the number of controllers. Finding the optimal placement S is referred to as the *minimum k-median* problem in the literature and known to be NP-hard [31]. In [25], authors solve the minimum k-median problem via exhaustively searching the all possible combinations, which takes weeks to complete. In this work, we use the following integer-linear program (ILP)

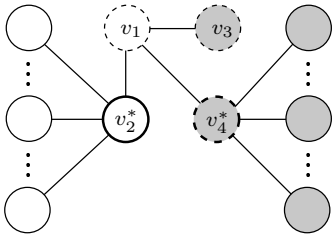


Figure 44: A sample network partition composed of two incontiguous parts.

to solve the minimum k-median problem.

$$\text{minimize} \quad \sum_{c \in V} \sum_{v \in V} z(c, v)\, d(c, v) \tag{1}$$

$$\text{subject to} \quad \sum_{c \in V} z(c, v) = 1, \quad \forall v \in V \tag{2}$$

$$z(c, v) \leq y(c), \quad \forall c, v \in V \tag{3}$$

$$\sum_{c \in V} y(c) = k \tag{4}$$

$$\text{where} \quad y(v) = \begin{cases} 1, & v \text{ is a median} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$z(c, v) = \begin{cases} 1, & c \text{ is a median for } v \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

The given program introduces two binary variables $y(v)$ and $z(c, v)$, where $y(v)$ is set 1 to indicate that $v \in V$ is chosen as a median and $z(c, v)$ is set 1 to indicate that v is connected to the median c. The optimization objective (1) minimizes the sum of the propagation latencies. Note that since $d(c, v)$ is constant, the optimization objective will try to connect each vertex v to its nearest median c. Constraint (2) assures that each vertex is connected to one and only one median. Inequality (3) constrains $z(c, v)$ to be turned on as long as c is chosen as a median, that is, $y(c)$ is 1. Finally, equality (4) constrains the program to select just k medians.

In terms of computational complexity, the integer program has $|V|+|V|^2$ variables, $|V| + |V|^2 + 1$ constraints and a single optimization objective.

Note that while the presented integer program solves the k-median problem, it

does not handle the cases, where the resultant parts are incontiguous, which is of significant importance in terms of partitioning networks. An example for this issue is depicted in Fig. 44. Here the network is partitioned into two parts represented in white and gray circles and the medians are v_2^* and v_4^* denoted in thick circles. While vertex v_3 has the same distance (i.e., propagation latency) to both medians, the integer program produced a solution that connects v_3 to v_4^*. In this case while the k-median problem constraints are satisfied, such an incontiguous partitioning would not be preferred by a network administrator. That being said, these corner cases can easily be spotted and resolved by a linear-time post-processing task without introducing a separate constraint to the integer program.

4.3.5 Routing Algorithms

In order to capture the effect of network partitioning on routing, we evaluate the performance results for the following four different routing algorithms.

First, we implement the shortest path (SP) routing on a network with unit link costs. In this scheme (SP-U), the goal is to find a path with the minimum number of hops. Next, we enhanced the algorithm to – instead of using unit link costs – take the link bandwidth usage into account. In this second scheme (SP-BW), the algorithm finds a path, where the sum of the costs of the links along the path are minimized. Since the cost of a path is actually dominated by the link with the maximum load, we further enhanced the second scheme to operate on a graph, where the link costs are set to their L2 norms. This enhancement (SP-NBW) conveys a strategy such that the weight of the highly loaded links on a path are amplified. Finally, we implement the minimax (aka. widest-path) routing algorithm, where the cost of a path is determined by the maximum link cost along the path. A list of optimization objectives for the described routing algorithms are given in Table 37. Here, p and $b(i, j)$ denotes a path and the bandwidth usage along the link (i, j), respectively.

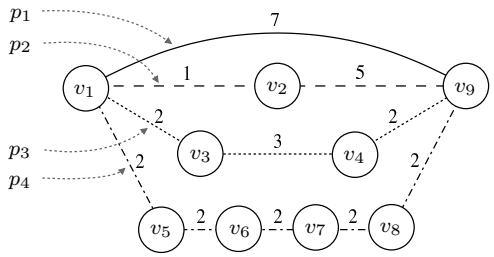


Figure 45: A sample graph for examining the routing algorithms. Edge costs denote the bandwidth usage along the links.

Routing Alg.	Optimization Obj.	Path Costs			
		p_1	p_2	p_3	p_4
SP-U	$\sum_{(i,j)\in p} 1$	**1**	2	3	5
SP-BW	$\sum_{(i,j)\in p} b(i,j)$	7	**6**	7	10
SP-NBW	$\sum_{(i,j)\in p} b^2(i,j)$	49	26	**17**	20
Minimax	$\max_{(i,j)\in p} b(i,j)$	7	5	3	**2**

Table 37: Optimization objectives for various routing algorithms and computed costs of the paths given in Figure 45.

In Figure 45, a sample graph is depicted to examine the routing algorithms for a flow from v_1 to v_9 with 4 alternative paths (i.e., p_1, p_2, p_3 and p_4). Here, edge costs denote the bandwidth usage along the links. Each routing algorithms picks the path with the minimum cost computed using the optimization objectives presented in Table 37, where costs of the available paths for different routing algorithms are provided for comparison. Here, algorithms make different optimal route decisions (marked in bold) determined by the corresponding optimization objective.

For the shortest path calculations, we set the controllers to implement Floyd-Warshall [32] all-pairs-shortest-path algorithm, which has a computation complexity of $O(V^3)$. In case of minimax, we use an iterative algorithm, where the links in the graph are removed iteratively in decreasing link cost order until the source and the destination nodes get disconnected. Internally, at each iteration, we peel off the link with the maximum cost and perform a breadth-first search to check the connectivity, which results in a complexity of $O(E^2)$. In conclusion, the computational complexity of the employed minimax algorithm adds up to $O(VE^2)$.

4.3.6 Cross-Traffic Generation

During each test period, after building the switch network and establishing its connectivity with the domain controllers, we generate artificial cross-traffic across hosts to stress the data and control plane capacities. In addition, we utilize the data latency and throughput statistics populated by the cross- traffic generator to evaluate the performance of the system for the configured setting.

Cross-traffic generator system is composed of client and server pairs written in Python. Each cross-client emulates a stream of flows that incorporates the presented statistical models in [33] and [34]. Here, the flow arrival time is exponentially distributed ($\lambda = 1$) and 80% of the flows last less than 10 seconds. The destination host for each flow is selected uniformly random among all available hosts. Flows are transmitted over TCP sockets using a best-effort approach in buffers of size 128 KB. In order to avoid repeating the same congestion patterns and to further exploit the routing algorithms taking link costs into account, we dynamically create hotspots in the network. That is, 30% of the clients are randomly chosen to act as an hotspot and perform 3x more flows compared to a regular client. Hotspot assignments are scheduled to be changed periodically every 10 seconds.

Cross-traffic servers are set to measure the data throughput and latency over the received datagrams. Further, servers are initially provided the network map and the switch partition. Hence, upon retrieval of a datagram, they can figure out if the data originates from an external domain and provide statistics for both internal and external flows individually.

4.3.7 Performance Metrics

In order to have a comprehensive performance evaluation of the conducted experiments, we employ several agents to observe the data and control plane statistics throughout the tests.

The fundamental measures of the data plane performance are provided by the flow latency and throughput metrics. We employ a set of applications in the cross-traffic generator system to observe the flow latency and throughput over the received datagrams and produce a report of the populated statistics before the exit.

In order to measure the flow setup time performance of both data and control planes, we run a setup time measurement agent on each host that periodically tries to create a new flow towards all available hosts and measures the end-to-end flow establishment time. Note that the setup time measurement agent is set to create flows such that it is ensured that no suitable rules are present in the switches and, consequently, the control plane is forced to populate new rules for the incoming flows.

Each switch incurs a certain amount of control plane overhead due to the generated flows. Hence, the number of switches connected to a controller has a direct impact on the control plane performance. In relation with the partition size, as the number of domains increase, the number of switches contained by a domain and the control plane load imposed by the switches decrease. That being said, the increase in the number of domains (i.e., controllers) also drives the inter-controller messaging volume upwards. In order to encapsulate both facets of partitiong on control plane performance, we provide statistics for the PacketIn and FlowMod messages processed by controllers and the inter-controller messaging volume.

4.4 Experimental Results

In Section 4.3, we presented the details of the used algorithms and discussed the design choices employed throughout the test framework. In this section, we will present and discuss the properties of the used data set and the empirical results of the conducted experiments. In the overall picture, we evaluate the flow performance for a set of metrics on a test bed such that different network topologies are employed with a multitude of partitions. In the conducted experiments, we stress each network

Table 38: Properties of the network topologies.

Name	$\|V\|$	$\|E\|$	$d_{avg}(E)$
BtNorthAmerica	36	76	3.0
Dfn	58	87	3.0
Garr201004	54	58	2.52
HiberniaGlobal	55	81	2.95
Switch	54	68	2.52

topology and partition configuration using 4 different routing engines (that is, shortest path with unit and link load edge weights, shortest path with normalized link loads, and minimax) and 3 different inter-controller topology update periods (10, 30, and 60 seconds). Each configuration is evaluated for a period of 5 minutes. In addition, we repeat each test for 9 times.

4.4.1 Network Topology

In the conducted experiments, we used five representative network topologies (BtNorthAmerica, Dfn, Garr201004, HiberniaGlobal and Switch) selected from the Internet Topology Zoo project [35]. Properties of the used topologies are presented in Table 38. Here $|V|$ denotes the number of vertices, $|E|$ denotes the number of edges and $d_{avg}(E)$ denotes the average vertex degree of the graph. In Table 38, topologies BtNorthAmerica and Dfn are chosen due to their high average vertex degrees. That is, link load adaptive routing algorithms (e.g., Minimax) will be provided a sufficient number of multiple paths for making alternative routing decisions. In the tests, we found out that the switch to controller connections in the topologies larger than 60 vertices start to experience timeouts due to the system load imposed by the data plane processing. Switch timeouts further yield uncertain fluctuations in the test results. Hence, we picked topologies containing less than 60 vertices.

In order to further investigate the properties of the employed network topologies, in Figure 46, we present CDF plots of the vertex degrees and the shortest path lengths in switch-hops such that the links are of unit cost. Here, BtNorthAmerica takes the

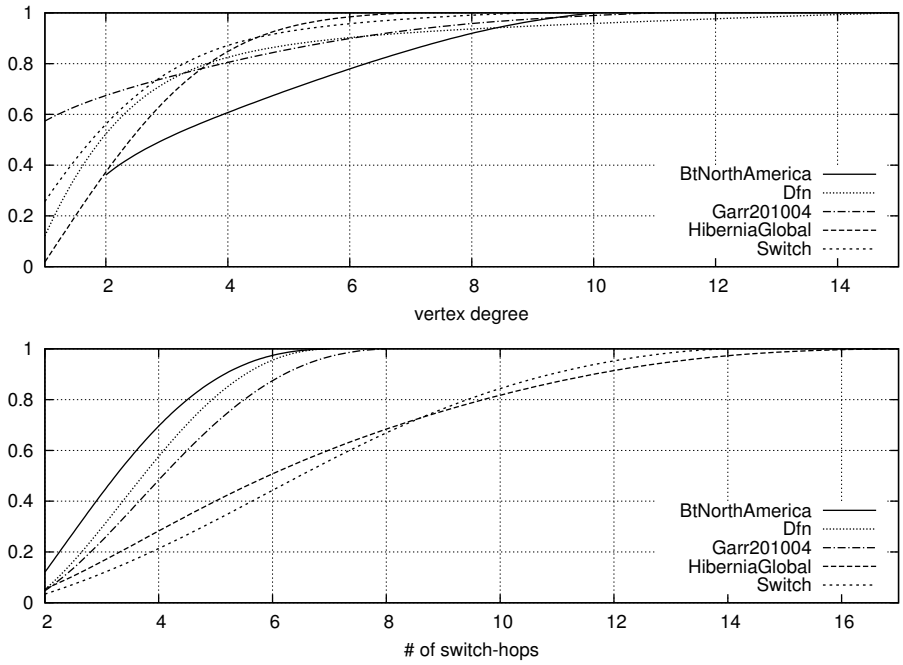


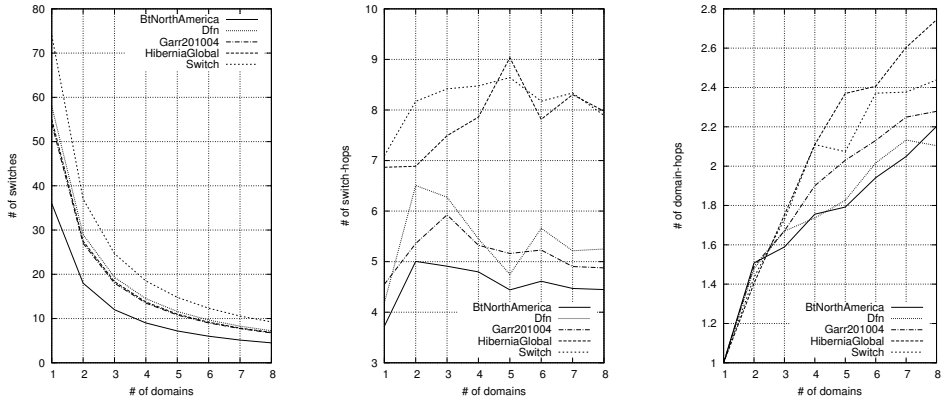
Figure 46: CDF of the vertex degrees and the shortest path lengths (in switch-hops) of the used topologies.

lead compared to the other graphs in terms of highest vertex degree. That is, almost half of the vertices in BtNorthAmerica has at least 3 edges. As a result of this, it is anticipated that the link load adaptive routing algorithms will perform better in BtNorthAmerica. Further, HiberniaGlobal and Switch topologies has longer shortest paths, hence, these topologies are expected to incur higher packet latencies in the tests.

4.4.2 Partitioning Results

In the conducted experiments, we partitioned the network topologies using the average-case latency metric described in Section 4.3.4. Properties of the resultant partitions are shown in Figure 47.

In Figure 47a, the average number of vertices contained by domains for different partitions are given. Here as the number of domains increases, the average domain size (i.e., the number of switches connected to a controller) decreases. As a result of



(a) The average number of vertices contained by parts.

(b) The average length (in switch-hops) of shortest paths.

(c) The average number of shortest path lengths in domain-hops.

Figure 47: Partitioning results of the used network topologies.

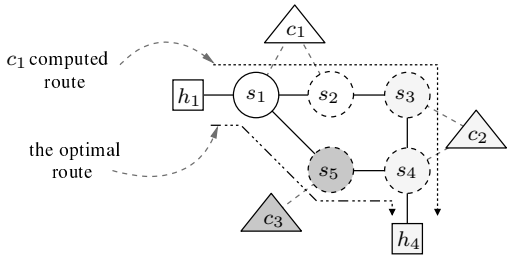


Figure 48: A sample network partition, where c_1 routes the flow $h_1 \rightarrow h_4$ using $[s_1, s_2, s_3, s_4]$ path (whereas, $[s_1, s_5, s_4]$ is a shorter alternative) due to the partially exposed internal network information by the controllers.

this, the load on the controller imposed by the switches is anticipated to decrease as well.

In Figure 47b, the average length of the shortest paths, where the edges are of unit cost, are presented in switch-hops. Note that the controllers in the system expose just the costs of the inter-edge-switch routes in their domains. Controllers use this broadcasted information to perform routing decisions. Due to the fact that the controllers do not know the internals of an external network controlled by a remote controller, they might not always come up with the optimal route. An example of such a misrouting decision is depicted in Figure 48. Here, in a network composed of 3 domains, c_1 is to make a routing decision for flow $h_1 \rightarrow h_4$. Given that c_1 is only provided the information of domain reachability and inter-edge-switch costs of

domains, it does have no prior knowledge of either the network internals, or the attachment point of h_4 in c_3's network. Hence, moving from the fact that c_1 is directly adjacent to c_2 and routing a packet to c_2 over c_3 appears like adding just an extra hop to the eventual path, c_1 prefers to route the flow directly over $[s_1, s_2, s_3, s_4]$ path, whereas the $[s_1, s_5, s_4]$ path passing over c_3's network is a shorter alternative. On the other hand, the exposed internal network information is anticipated to be more fine-grained as the partition sizes increase, which eventually would result in better routing decisions. This effect of the partitioning is observable in Figure 47b. Here, in more coarsed-grained partitions (e.g., partitions of size 2 and 3), the average of the shortest path lengths tends to increase. Further, as the partitioning gets more fine-grained, the average of the shortest path lengths starts to decrease. Note that there are two optimal configuration alternatives for minimizing the average of the shortest path lengths: 1) a single domain network, and 2) a network, where each switch is assigned to an individual domain.

In Figure 47c, the average length of the shortest paths in domain-hops are shown. Note that an external flow (that is, a flow passing through multiple domains) incur extra flow setup time cost due to the necessary amount of control plane processing and switch programming individually performed by each domain controller. That being said, while the number of domains a typical flow would pass is anticipated to increase proportional to the number of domains in the network partition, in Figure 47c, it is shown that this cost factor becomes diminishing due to the small number of domain-hops. That is, even when the network is divided into 8 parts, the average length of the shortest paths in domain-hops is still under 3.

4.4.3 Cross-Traffic Patterns

In order to fix the generated cross-traffic constant across individual tests, we set each client to replay a pre-determined list of flows – including the mobile hotspots, which

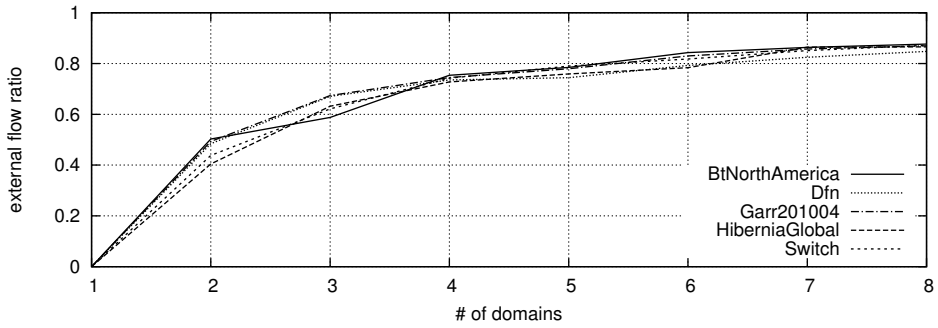


Figure 49: Ratio of the external (i.e., inter-domain) flows for different partitions.

are purposed to congest random regions in the network periodically. Due to the fact that the network topologies generate different routes for flows under different partition configurations, the ratio of the external flows vary with the employed graph and the partition. This implication of partitioning is presented in Figure 49. Here, the ratio of the external flows develops proportional to the number of domains. That is, while the average of the external flows consitute the 46% of the entire flow space for $k = 2$ (i.e., where there are two domains), it mounts upto 86% for $k = 8$. Put another way, external flows are anticipated the dominate the performance results as the domains increase in numbers.

4.4.4 Control Plane Load Results

In order to quantify the effect of the network partitioning on the control plane load, we instrumented the control plane to observe the following two variables. First, we employed a set of counters on the controllers for tracking the number of *processed* messages of type PacketIn (that is, the messages sent by the data plane to the controller for unmatched flows) and FlowMod (that is, the messages sent by the control plane to the switches to orchestrate the data plane for routing a particular flow). Second, we collect the total volume of the broadcasted inter-controller messages (that is, the IGMP traffic), which we previously stated that it is anticipated to increase due to the encapsulation of the more fine-grained information on the network internals

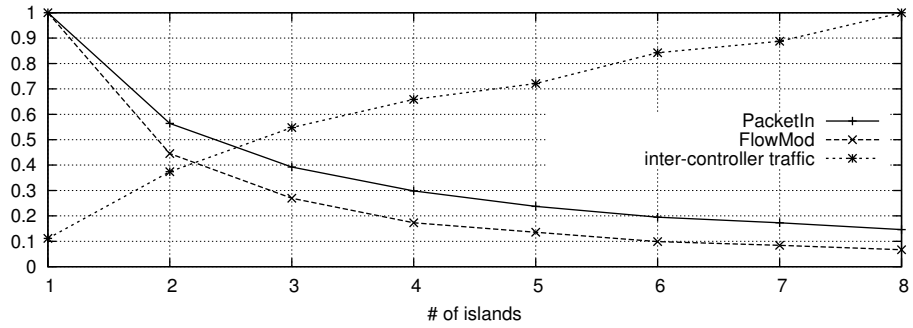


Figure 50: Normalized control plane performance results.

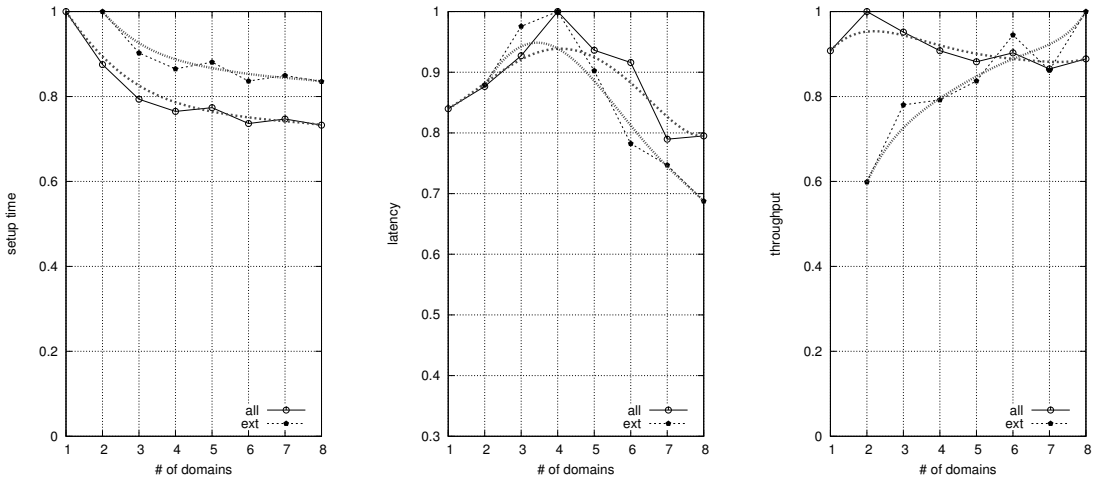
as the domains become smaller in size. The results depicting these observations are presented in Figure 50.

In the results, we observe an exponential decrease in the number of received PacketIn and sent FlowMod messages. That is, the results decline down to 71% and 83% at $k = 4$ for processed messages of types PacketIn and FlowMod, respectively, and further falls back to 86% and 94% at $k = 8$. In other words, the number of domains has a significant effect on the control plane packet processing load.

The employed architecture operates on an out-of-band control plane, that is, data and control planes are operated by two distinct networks. That being said, we further present the total amount of broadcasted inter-controller messages to observe the effect of partitioning on the control plane traffic. Due to the employed IGMP communication, the inter-controller traffic follows a linear increase with the number of parts. (See Figure 50.) If peers would have employed a peer-to-peer communication model, this figure would follow an exponential path. That being said, the actual maximum magnitude of the inter-controller message traffic is not much more than 80 kbps, which is quite negligible.

4.4.5 Flow Setup Time Results

Due to the employed reactive programming model, each controller reacts individually to a flow passing across multiple domains. As a result of this, the average setup



(a) Flow setup time results. (b) Flow latency results. (c) Flow throughput results.

Figure 51: Normalized flow setup time, latency and throughput performance results.

time performance of flows is dominated by the following three key factors: 1) the controller loads and the number of 2) switches and 3) domains a flow passes through. First, Section 4.4.4 presents that the controllers experience an exponential decrease in received messages, which have a direct impact on the contoller loads. Second, Figure 47b depicts that the partitioning does not induce critical misrouting decisions and falls short of having a significant impact in the overall picture. Third, Figure 47c presents that the number of domains a flow passes through follows a logarithmic growth as the domain sizes decrease linearly. That is, a typical flow is anticipated to pass through approximately 2 domains for $k = 4$ (that is, partitions of size 4) and 2.5 domains for $k = 8$. Putting it all together, while the number of domains a flow passes grows logarithmicly, the controller loads cut down exponentially. Hence, the extra flow setup delay contributions due to the involvement of an increasing number of controllers remain limited and the response time of the controllers start to dominate the setup time of the flows. Consequently, as the domains shrink and controllers experience less load, flows are anticipated to observe a decrease in flow setup times. Indeed, this observation holds with the performance figures, even for external flows.

In Figure 51a, we present the normalized flow setup time (i.e., the round-trip-time of a flow that does not match with any of the existing rules on the switches) performance results. Here, lines denote the average of the samples collected from a multitude of configurations (i.e., retry, topology, routing algorithm, update period). Figure depicts the averages for external and all (i.e., both internal and external) types of flows.

Figure 51a depicts an improvement up to 24% at $k = 4$ for all flows. For partitions of $k \geq 4$, the improvement starts to stall and stabilize. (Standard deviation is 8.5%.) Figure 47a shows that the employed partitions contain 15 switches in each domain on the average for $k = 4$. This concludes that the generated cross-traffic start to overload the controllers for networks composed of 15 switches or more – given the physical computation resources of the test environment.

Since controllers react individually and sequentially to external flows, flows passing across multiple domains observe the highest setup time impact from the partitioning. In addition, as shown in Figure 49, the ratio of the external flows increase upto 80% as the number of domains increase. Put another way, external flows start to dominate the overall setup time performance as the number of domains mounts up. That being said, contrary to the common anticipation, Figure 51a shows that the external (i.e., inter-domain) flows experience a flow setup time decrease around 14% at $k = 4$ and 17% at $k = 8$. (Standard deviation is 5.8%.) These results holds with the previously stated observation that the controller loads are the predominant driver of the flow setup times.

4.4.6 Flow Latency & Throughput Results

While controller loads constitute the major impact factor in terms of flow setup time performance, latency and throughput performance is dominated by the quality of the paths chosen. That is, the number of switches along a path and the link states (e.g.,

available bandwidth, transmission rate) encapsulate the principal determinants for flow latency and throughput. In Figure 48, an example network topology demonstrating the effect of partitioning on route selection is given. Indeed, Figure 47b depicts the changes in routing quality implied by the underlying partition. In addition, routing algorithms utilizing link states (i.e., SP-BW, SP-NBW, and Minimax) would be provided more fine-grained network information and are anticipated to come up with higher quality routes as domains increase in numbers. As a matter of fact, the optimal latency and throughput figures are achieved when either there is a single domain or each switch assigned to an individual domain. Hence, the observed performance is expected to degrade for coarse-grained partitions and later improve as the domains shrink in size.

In the conducted tests, the flow latency and throughput performances are computed by the cross-traffic servers via comparing the timestamp of the received datagrams with the system time and dividing the flow size to the total amount of time it takes to receive the flow, respectively. Accordingly, the results represent the end-to-end packet latency and data transmission throughput of flows. The normalized means of the collected samples from these measurements are presented in Figures 51b and 51c.

Latency For all types of flows including internal and external, Figure 51b depicts a flow latency performance degredation upto 17% for $k = 4$, which later retracts towards $k = 7$ and catches the performance level given at $k = 1$. (Standard deviation is 6.8%.) On the other hand, while external flows follow a similar latency increase pattern for $k = 3$ and $k = 4$, it catches the initial condition at $k = 6$ and gets better as domains increase in size. (Standard deviation is 10%.) These findings holds with the previously stated expectation that the shrinkage in domain sizes reveals more fine-grained information in the broadcasted network traffic messages and eventually leads

to improved inter-domain route calculations close to the optimal. Consequently, due to the lack of necessary amount of information, controllers employed in the coarse-grained partitions (e.g., $k = 3$ or $k = 4$) might produce routes passing over congested links in the network, which ends up with higher packet latency.

Throughput In Figure 51c, the overall flow throughput stays more or less constant, where the standard deviation is 4%. On the other hand, external flows observe an increase in throughput performance upto 24% at $k = 4$ and 41% at $k = 8$. (Standard deviation is 11%.) External throughput figures hold with the fact that the decrease in domain sizes lets controllers to perform enhanced routing decisions, which eventually leads to improved flow throughput.

In the summary, overall flow latency and throughput performance observe minor changes after partitioning. Further, the increasing number of external flows are subject to improved routing decisions by controllers as the exposed network information develops with the growing number of domains.

4.5 Related Work

Software-defined networking literature investigating the reliability and scalability issues in multi-domain architectures can be grouped under the following three approaches: 1) those who trade fine-grained flow-level visibility in the control plane with scalability, 2) analytical models, and 3) distributed controller implementations.

Early studies present approaches such that a portion of the control plane functionality is pushed towards the switches in the data plane for the purpose of improving the scability. [36] partitions and delegates the control plane policies to a set of authority switches in the network. Here switches forward unmatched flows to the authority switches, which helps to mitigate the load on the control plane. [37] decentralizes the control plane by pushing the management of flows that require state information at short time scales to the controllers placed next to the switches. [38] purposes

TCP buffer sizes at the end hosts to detect and mark the elephant flows. Consequently, the central controller installs specifically tailored rules for these flows to the switches to improve the overall network performance. [39] presents a platform, where the short-lived flows are handled in the data path and the rest is forwarded to the controller.

A group of publications in the literature analytically investigate the effect of various variables in a network architecture connected to multiple controllers. [40] studies how to optimally partition a network into a given number of intersecting domains such that the average number of domains a flow travels is minimized. Here they presume that each domain is managed by a local controller. [25] investigates the effect of the controller placement on the switch-controller latency. They further examine the implications on mesh restoration and ring protection. [41] demonstrates a network calculus-based analysis on the performance of a hierarchical SDN architecture deployment shown in [42]. [43] studies the application of local algorithms from distributed computing literature for both hierarchical and flat multi-domain network deployments. [44] develops an analytical model such that the switches in the network is dynamically assigned to a set of controllers and the optimization objective is to minimize the flow setup time and communication overhead. [45] investigates the reliability-aware controller placement problem, where the reliability is defined in terms of the number of broken control paths due to network failures.

There are further studies in the literature, where the authors present their results for various multiple controller settings verified by custom controller implementations. [18] presents a distributed control plane such that a multitude of controllers are in charge of certain regions of the network and perform passive state synchronization via an event bus. [22] proposes a forwarding service, where controllers placed next to top-of-rack switches re-write ethernet fields that encapsulate Valiant load-balanced

source routes using in-packet Bloom filters. [20] presents a distributed controller architecture programming interface upon which developers can build a wide variety of management applications. They present a comprehensive list of issues inherent in distributed controller deployments and discuss existing solutions to address these concerns. [19] presents simulation results that investigates the effect of state synchronization frequency between two controllers on the performance of a load balancer application. [21] presents an implementation of a distributed controller architecture, where the control plane capacity can be dynamically increased or decreased without any changes to the existing OpenFlow protocol. They further present experimental results on the performance of migrating a switch from one controller to another and the increase in the data plane performance proportional to the number of available controllers. [42] introduces a hierarchical controller model, where a set of local controllers and a root controller orchestrate the network jointly. In this work, local applications (e.g., elephant flow detection, link discovery) are handled by a local controller within the vicinity of a switch and the rest is delegated to the root. [46] purposes a set of load balancers to interface the switches with a cluster of controllers. An unmatched flow packet is first forwarded to the load balancer in charge of the domain and then the load balancer hands the packet off to an available controller. [17] presents the architectural details and empirical analysis of a massive scale software-defined WAN deployment based on the work given in [20]. The employed network runs jointly with existing routing protocols (e.g., BGP, IS-IS, OSPF) and provides versatile traffic engineering (TE) and resilience capabilities. They also present experienced TE operation and link load utilization statistics of the architeture. [47] outlines a distributed controller model such that the controllers exchange reachability information to establish multi-domain routes. That being said, the work lacks any empirical results. [48] proposes a novel switch migration protocol for enabling an elastic distributed controller. [49] provides an open distributed control plane for

multi-domain networks based on a unique message-oriented communication bus. It features agents for adaptive monitoring with regards to security restrictions, low bandwidth inter-connections, end-to-end QoS, and mobility management. [50] investiages the high-availability of the controllers in OpenFlow switch specification version 1.2+ and proposes a redundant scheme to tackle both single- and multi-domain recovery scenarios. In a recent study, [51] presents an experimental distributed SDN control platform motivated by the performance, scalability, and availability requirements of large operator networks.

CHAPTER V

CONCLUSION

The wireless Internet is experiencing tremendous growth thanks to the introduction of smart phones and associated bandwidth-hungry applications. The MSPs are struggling to keep up with this demand in today's networks. The 5G architecture needs to bring a high capacity, agile, low cost solution to ensure both user satisfaction and MSP profitability. This dissertation introduced a programmable, all-SDN architecture with hierarchical network control capabilities to allow different grades of performance for all fundamental network functionalities.

Using the proposed all-SDN architecture, we introduced a unified approach to mobility, handoff and routing management propose Connectivity Management as a Service (CMaaS). CMaaS allows the control of mobility and routing of different flows or users differently in the network thereby opening a new revenue generation path to MSPs.

In order to investigate the realization of an all-SDN architecture, we investigated the flow-level performance of multi-domain networks orchestrated by multiple controllers and discussed the results obtained from a diverse set of configurations emulated using a custom software-defined networking architecture implementation. First, we presented an integer-linear programming model for the partitioning problem, discussed the implications of partitioning on routing, and conducted experiments using a multitude of configurations (i.e., network topologies, routing algorithms, link state update periods). Next, we presented our findings obtained from an exhaustive set of configurations.

Bibliography

[1] "Network Functions Virtualization (NFV): Use Cases," tech. rep., October 2013.

[2] C. Kulin and D. Ran, "C-RAN: The Road Towards Green RAN," October 2011.

[3] Kempf, J. and Johansson, B. and Pettersson, S. and Luning, H. and Nilsson, T., "Moving the mobile Evolved Packet Core to the cloud," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on*, pp. 784–791, October 2012.

[4] H. Holma and A. Toskala, *LTE for UMTS: Evolution to LTE-Advanced, 2nd Edition*. John Wiley & Sons, Inc., 2011.

[5] 3GPP, "Overview of 3GPP Release 12," tech. rep., June 2014.

[6] X. Lin, J. Andrews, A. Ghosh, and R. Ratasuk, "An overview of 3GPP device-to-device proximity services," *Communications Magazine, IEEE*, vol. 52, pp. 40–48, April 2014.

[7] O. N. Foundation, "OpenFlow Switch Specification," tech. rep., October 2013.

[8] K. Pentikousis, Y. Wang, and W. Hu, "MobileFlow: Toward software-defined mobile networks," *Communications Magazine, IEEE*, vol. 51, pp. 44–53, July 2013.

[9] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "SoftCell: Scalable and Flexible Cellular Core Network Architecture," in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, (New York, NY, USA), pp. 163–174, ACM, 2013.

[10] K.-K. Yap, R. Sherwood, M. Kobayashi, T.-Y. Huang, M. Chan, N. Handigol, N. McKeown, and G. Parulkar, "Blueprint for Introducing Innovation into Wireless Mobile Networks," in *Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, VISA '10, (New York, NY, USA), pp. 25–32, ACM, 2010.

[11] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "OpenRadio: A Programmable Wireless Dataplane," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, (New York, NY, USA), pp. 109–114, ACM, 2012.

[12] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "SoftRAN: Software Defined Radio Access Network," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, (New York, NY, USA), pp. 25–30, ACM, 2013.

[13] S. Kumar, D. Cifuentes, S. Gollakota, and D. Katabi, "Bringing Cross-layer MIMO to Today's Wireless LANs," *SIGCOMM Comput. Commun. Rev.*, vol. 43, pp. 387–398, Aug. 2013.

[14] A. Racz, A. Temesvary, and N. Reider, "Handover Performance in 3GPP Long Term Evolution (LTE) Systems," in *Mobile and Wireless Communications Summit, 2007. 16th IST*, pp. 1–5, July 2007.

[15] D. Erickson, "The Beacon Openflow Controller," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, (New York, NY, USA), pp. 13–18, ACM, 2013.

[16] R. R. Scarabucci, M. A. Stanton, M. R. X. d. Barros, M. R. Salvador, S. M. Rossi, F. D. Simoes, M. L. Rocha, I. L. d. S. Neto, J. B. Rosolem, T. R. T. Fudoli, J. M. D. Mendes, N. F. Castro, I. Machado, A. E. Reggiani, A. Paradisi, and L. Martins, "Project GIGA-High-Speed Experimental IP/WDM Network," in *Proceedings of the 1st International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*, TRIDENTCOM '05, (Washington, DC, USA), pp. 242–251, IEEE Computer Society, 2005.

[17] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a Globally-Deployed Software Defined WAN," *SIGCOMM Comput. Commun. Rev.*, vol. 43, pp. 3–14, Aug. 2013.

[18] A. Tootoonchian and Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow," in *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, INM/WREN'10, (Berkeley, CA, USA), pp. 3–3, USENIX Association, 2010.

[19] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically Centralized?: State Distribution Trade-offs in Software Defined Networks," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, (New York, NY, USA), pp. 1–6, ACM, 2012.

[20] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A Distributed Control Platform for Large-scale Production Networks," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, (Berkeley, CA, USA), pp. 1–6, USENIX Association, 2010.

[21] V. Yazici, M. O. Sunay, and A. O. Ercan, "Controlling a Software-Defined Network via Distributed Controllers," in *NEM Summit*, (İstanbul, Turkey), October 2012.

[22] C. Macapuna, C. Rothenberg, and M. Magalhaes, "In-packet Bloom filter based data center networking with distributed OpenFlow controllers," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pp. 584–588, Dec 2010.

[23] AT&T, "Vision Alignment Challenge Technology Survey," tech. rep., Nov 2013.

[24] NTT Communications, "12 Months of SDN Innovation and Deployments," Mar 2014.

[25] B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement Problem," in *Proceedings of the 1st Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, (New York, NY, USA), pp. 7–12, ACM, 2012.

[26] B. Heller, *Reproducible network research with high-fidelity emulation*. PhD thesis, Stanford University, 2013.

[27] "Open vSwitch," January 2014.

[28] "Floodlight OpenFlow Controller," January 2014.

[29] H. Holbrook, B. Cain, and B. Haberman, "Internet Group Management Protocol Version 3 (IGMPv3)," Aug 2006.

[30] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Mar. 2008.

[31] K. Jain and V. V. Vazirani, "Approximation Algorithms for Metric Facility Location and k-Median Problems Using the Primal-dual Schema and Lagrangian Relaxation," *J. ACM*, vol. 48, pp. 274–296, Mar. 2001.

[32] R. W. Floyd, "Algorithm 97: Shortest Path," *Commun. ACM*, vol. 5, pp. 345–, June 1962.

[33] T. Benson, A. Akella, and D. A. Maltz, "Network Traffic Characteristics of Data Centers in the Wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, (New York, NY, USA), pp. 267–280, ACM, 2010.

[34] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The Nature of Data Center Traffic: Measurements & Analysis," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, (New York, NY, USA), pp. 202–208, ACM, 2009.

[35] S. Knight, H. X. Nguyen, N. Falkner, R. A. Bowden, and M. Roughan, "The Internet Topology Zoo.," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[36] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. –, Aug. 2010.

[37] J.-Y. Yoo, N. Kim, and J. Kim, "Implementation of Decentralized Per-flow Control Based on Timescale Decomposition for Software Defined Networks," in *Proceedings of the 6th International Conference on Future Internet Technologies*, CFI '11, (New York, NY, USA), pp. 42–45, ACM, 2011.

[38] A. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in *INFOCOM, 2011 Proceedings IEEE*, pp. 1629–1637, April 2011.

[39] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling Flow Management for High-performance Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 254–265, Aug. 2011.

[40] A.-W. Tam, K. Xi, and H. Chao, "Use of Devolved Controllers in Data Center Networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pp. 596–601, April 2011.

[41] S. Azodolmolky, P. Wieder, and R. Yahyapour, "Performance Evaluation of a Scalable Software-Defined Networking Deployment," in *Proceedings of the 2nd European Workshop on Software Defined Networks*, EWSDN '13, Oct 2013.

[42] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, (New York, NY, USA), pp. 19–24, ACM, 2012.

[43] S. Schmid and J. Suomela, "Exploiting Locality in Distributed SDN Control," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, (New York, NY, USA), pp. 121–126, ACM, 2013.

[44] M. Bari, A. Roy, S. Chowdhury, Q. Zhang, M. Zhani, R. Ahmed, and R. Boutaba, "Dynamic Controller Provisioning in Software Defined Networks," in *Network and Service Management (CNSM), 2013 9th International Conference on*, pp. 18–25, Oct 2013.

[45] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware controller placement for Software-Defined Networks," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pp. 672–675, May 2013.

[46] P. Lin, J. Bi, and H. Hu, "ASIC: An Architecture for Scalable Intra-domain Control in OpenFlow," in *Proceedings of the 7th International Conference on Future Internet Technologies*, CFI '12, (New York, NY, USA), pp. 21–26, ACM, 2012.

[47] X. T. Phan, N. Thoai, and P. Kuonen, "A collaborative model for routing in multi-domains OpenFlow networks," in *Computing, Management and Telecommunications (ComManTel), 2013 International Conference on*, pp. 278–283, Jan 2013.

[48] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an Elastic Distributed SDN Controller," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, (New York, NY, USA), pp. 7–12, ACM, 2013.

[49] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed Multi-domain SDN Controllers," *CoRR*, vol. abs/1308.6138, 2013.

[50] K. Kuroki, N. Matsumoto, and M. Hayashi, "Scalable OpenFlow Controller Redundancy Tackling Local and Global Recoveries," in *Proceedings of the 5th International Conference on Advances in Future Internet*, AFIN '13, Aug 2013.

[51] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an Open, Distributed SDN OS," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, (New York, NY, USA), pp. 1–6, ACM, 2014.