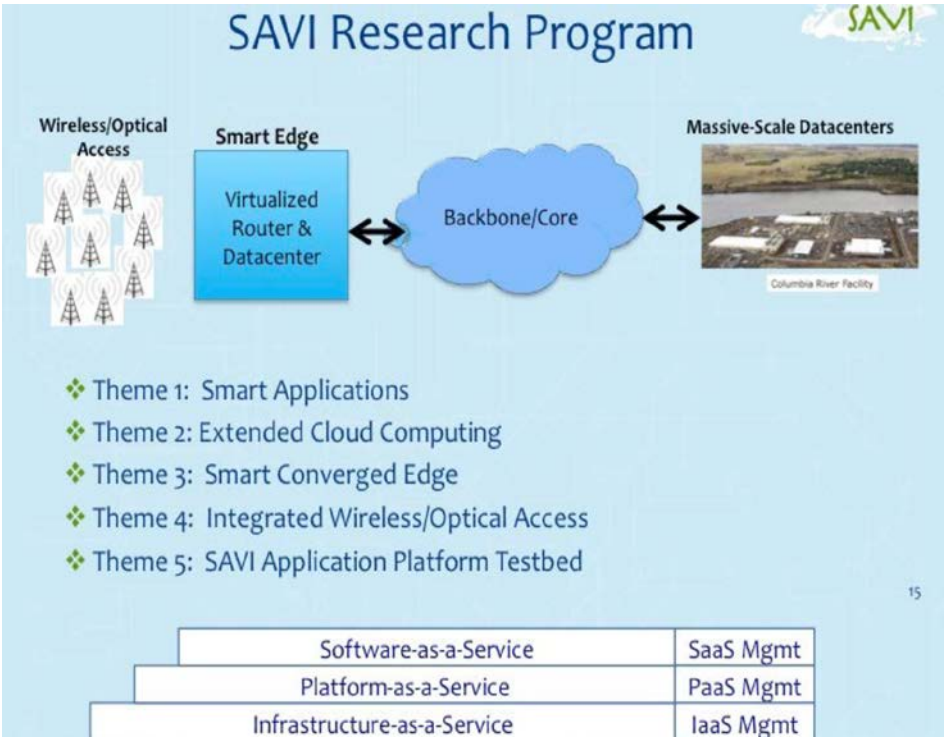
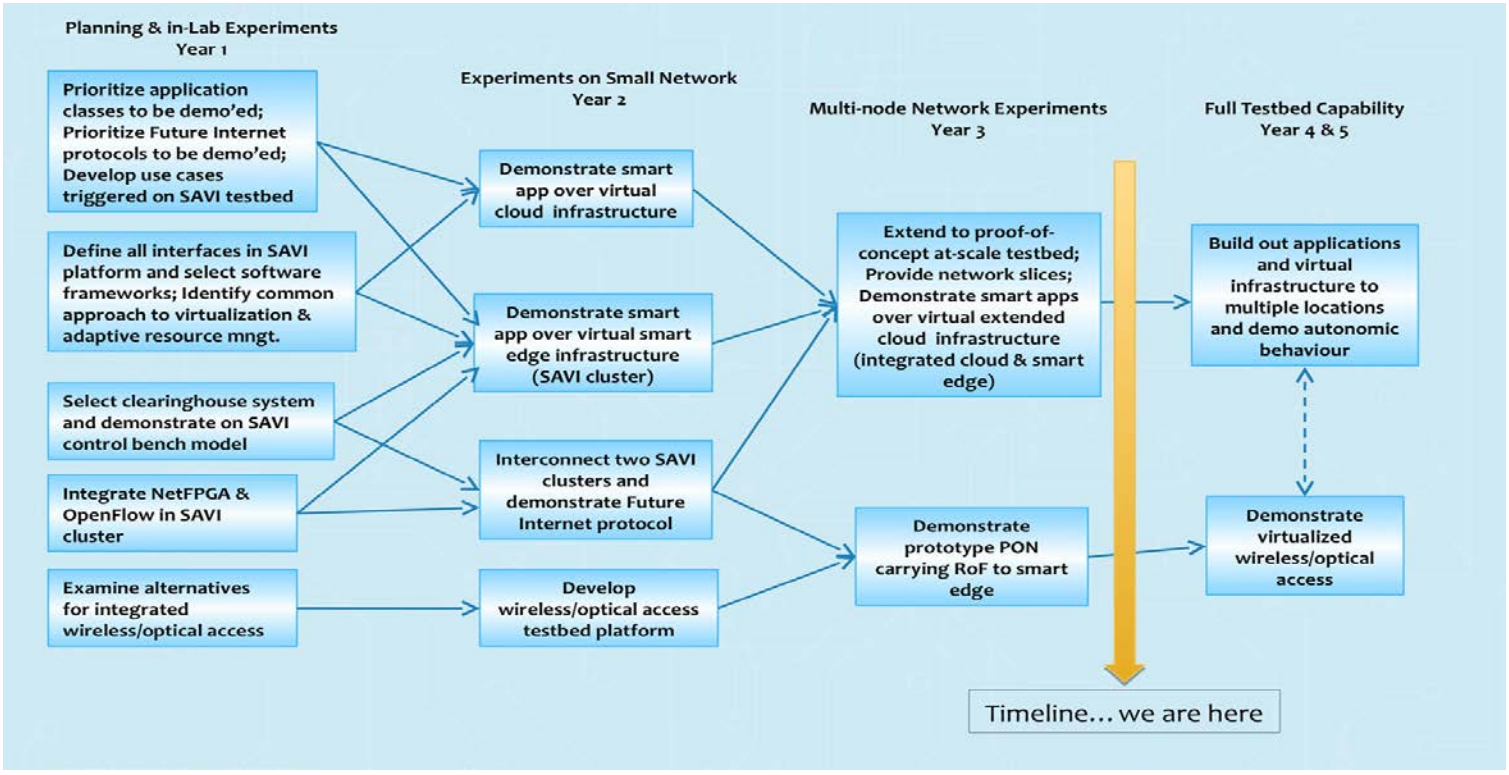
SAVI AGM'14 POSTER BOOKLET

This research has been funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Research (NSERC) Strategic Networks grant number NETGP 397724-10.





Opening Comments

SAVI 2014 Annual General Meeting (AGM): Student Posters

I am pleased to present the poster booklet derived from posters and demonstrations at the 2014 (Year 3) SAVI AGM held on the 7/8th of July. Over 100 people attended including SAVI Partners, SAVI Board members, Principal & Theme Leads, Researchers and students from many of the participating SAVI universities.

We are just about to reach the end of year three of SAVI, a year in which we planned and saw the integration of several important activities involving the multi-tier cloud structure, the creation of applications that run over both the edge and core cloud resources, the extensive use of software-defined infrastructure (SDI) and the introduction of wireless activities including small cells and the delivery of radio frequencies over optical fibre. This poster booklet demonstrates all of these research activities. As you work through these posters you will be able to see that that year 3 has brought together the disparate activities of years 1 and 2 into a more cohesive set of focussed activities.

I am also pleased at the levels of participation of researchers and the quality of posters this year. We have almost 30% more posters in year 3 than in year 2 and all major themes are comprehensively covered.

As we move into year 4, the SAVI research plan calls for us to focus our activities on: Teasing out the advantages of SDI that are made possible through a one-to-many core to edge infrastructure; and using this architecture to address the massive explosion in smart devices that within a few years are forecast to deliver data flows in multi-gigabit per second rates. With this in mind we have to drive into aspects of latency, network performance, and the impacts of small error rates on behaviours built into current monolithic networks. Network autonomic behaviour will be key in examining self-optimizing aspects of the smart edge.

I look forward to another year of innovation, experimentation and participation of all SAVI players.

Alberto Leon-Garcia
SAVI Scientific Director
July 2014

Towards Gigabit Smart Devices

As industry liaison and advisor to SAVI since its inception, I have been keeping an eye constantly on new advances being made within our ICT industry and can confidently say that SAVI has been a very early adopter of OpenFlow, OpenStack, Software-Defined Networking and the notions of multi-tier cloud as potential breakthrough architectures for future network deployments. In the first three years of SAVI, industry has gone from moderate interest to extreme interest with many innovations emerging around all of these technologies. Available market estimates indicate that the market for SDN enabled services will exceed $25B per annum by 2018. Additionally, the availability of R&D funds through venture capital alone into SDN focused companies has grown from $10m in 2009 to just over $500m in 2013. This is a fifty times growth of funding in four years. This industry spending and investment confirms that Software-Defined Networking is here to stay. The most important takeaway for SAVI is the need to keep ahead of the curve. What needs to be considered in staying ahead of the curve?

In my opinion the *EDGE* of the network and the relationships between *EDGE* and *CORE* structures will define the next generation of architectures. I hold this opinion because I see and read about the massive capacity expansion required within the next five years to satisfy data flows from smart devices that embody smart applications. The Wi-Fi Alliance already has in the market devices that can operate at three different frequencies (Tri-Band Devices at IEEE802.11ad) and these devices are capable of 7 Gbit/s data throughput. The Wi-Fi Alliance is already contemplating next generation devices at data rates of 30 Gbit/s for the 2020 time horizon.

In parallel with this, the telecommunication wireless giants are working diligently on 5G wireless networks with a focus on 1000-times capacity growth over todays 4G wireless networks. This growth is driven by the rapid evolution and population of smart devices, sustained emergence of machine-to-machine communications (including vehicles) and by the Internet of things. The most conservative of estimates indicate ~50B Internet-enabled devices by 2020. The focus of all this data production & consumption will be on the *EDGE* of the network with many forms of small to ultra-small dense cells that will embody architectures and virtualization strategies that will evolve over the next five years or so. In the Ericsson presentation at the 2014 AGM, Pierre Boucher talked at length about the challenges faced as these networks take shape. We should take heed of his comments not only about the massive scaling of the networks around us but also the socio-economic issues he addressed such as power efficiency and carbon footprint.

I believe in the SAVI concept and in particular two aspects, firstly: Software-Defined Infrastructure that enables both network infrastructure and compute resources to be seen as one, and secondly: Two tier Core and Edge deployment. In my view we will soon see the emergence of large-scale networks with massive deployment of edge capabilities to be the initial point of connection for the large number of data producing & consuming devices in the network of the future. This will be essential to manage latency and the impact of latency on device designs and the quality of experience the user will demand. SAVI has a critical role to play for Canada in ensuring that we can participate in this fundamental change in networks driven by real user experiences and device demand.

David Mann
SAVI Industry Liaison and Scientific Advisor
July 2014

We dedicate this year's poster booklet to the memory of

Professor Greg Steffan

Contents

This booklet is best viewed using Adobe Reader X or above.

Chapter 1

Theme 1 Smart Applications

Research Team

Baochun Li (Theme Lead), University of Toronto
Hausi Müller, University of Victoria
Eleni Stroulia, University of Alberta
Roch Glitho, Université Concordia University

1.1 Soda: A Case for Software-Defined Networking at the Application Layer

Younan Wang, **Baochun Li**, U. Toronto

Due to constraints on current SDN switches, existing solutions at the network layer may not be able to support traffic demand changes with a large number of flows. Catering to the needs of multi-party multimedia applications such as video conferencing, in this poster, we make a case for a new software-defined network architecture at the application layer.

We propose Soda, our new application-layer SDN architecture and traffic engineering solution, designed for the efficient utilization of the network capacity with a large number of flows, while maintaining utility max-min fairness. An important and salient advantage of Soda is its ability to support both real-time multimedia and traditional bulk transfer applications. Soda has been implemented and deployed in the Amazon EC2 inter-datacenter network, and its effectiveness has been evaluated with both simulations and real-world experiments.

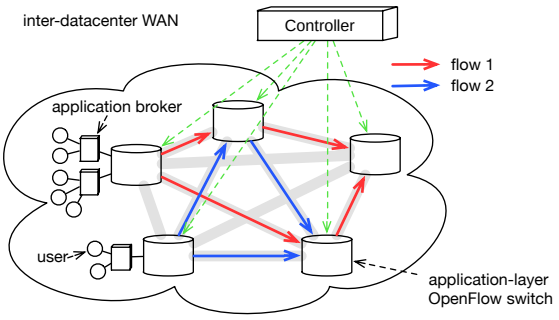
Soda: A Case for Software-Defined Networking at the Application Layer

Younan Wang, Baochun Li

Abstract

Due to constraints on current SDN switches (such as the rule count limit), existing solutions at the network layer may not be able to support traffic demand changes with a large number of flows. Catering to the needs of multi-party multimedia applications such as video conferencing, in this poster, we make a case for a new software-defined network architecture at the application layer. We propose Soda, our new application-layer SDN architecture and traffic engineering solution, designed for the efficient utilization of the network capacity with a large number of flows, while maintaining utility max-min fairness. A salient advantage of Soda is its ability to support both real-time multimedia and traditional bulk transfer applications. Soda has been implemented and deployed in the Amazon EC2 inter-datacenter network, and its effectiveness has been evaluated with both simulations and real-world experiments.

Overview of the Architectural Design

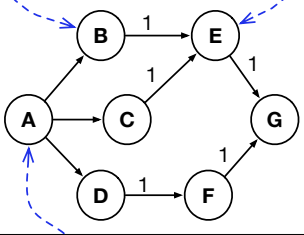


Integrate Pathlet Routing into Soda

Motivation: To realize source routing with a much smaller packet header size, compared to the traditional source routing scenario where all nodes along a path must be included in the packet header.

Flow ID	Next-Hop	Remaining FIDs
1	E	Empty

Flow ID	Next-Hop	Remaining FIDs
1	G	Empty



Group ID	Group Type	Buckets	Next-Hop	Remaining FIDs	Bucket Weight
1	SELECT	Bucket 1	B	Push(1,1)	10%
		Bucket 2	C	Push(1,1)	20%
		Bucket 3	D	Push(1,1)	70%

Utility Max-Min Fair Allocation

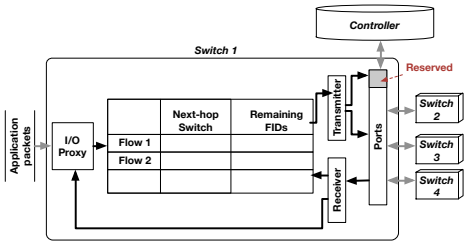
In the K-th iteration, we maximize the K-th smallest common utility and fix relevant flows:

$$\max \min_{i \in V} \phi_i\Big(\sum_{j \in P_i} b_{ij}\Big)$$

s.t.

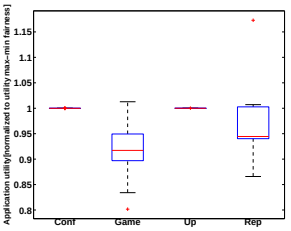
$$\begin{aligned}
\textstyle\sum_{j \in P_i} b_{ij} = \phi_i^{-1}(u_M^k), \quad & \forall k \in (1, \ldots, K-1), \\
& i \in B^k. \\
b_{ij} \geq 0, \quad & \forall j \in P_i,\ \forall i \in V. \\
\textstyle\sum_{j \in P_i} b_{ij} \leq D_i, \quad & \forall i \in V. \\
\phi_i\big(\textstyle\sum_{j \in P_i} b_{ij}\big) \text{ are equal}, \quad & \forall i \in V. \\
\textstyle\sum_{i \in I} \sum_{j \in P_i} b_{ij} \cdot \delta_{jl} \leq C_l, \quad & \forall l \in L.
\end{aligned}$$

Implementing an Application-Layer OpenFlow Switch

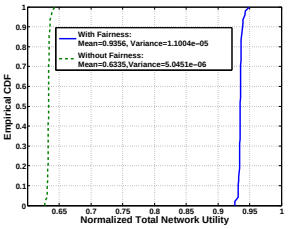


Performance Evaluation

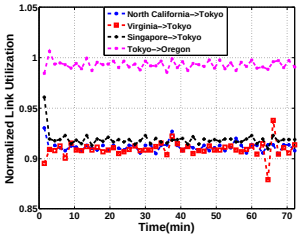
Fairness:



Network Utilization:



Link Utilization:



This work is funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP394424-10

FIGURE 1.1: *Soda: A Case for Software-Defined Networking at the Application Layer from Younan Wang and* ***Baochun Li*** *at University of Toronto (Theme 1)*

1.2 Consistency Models in Distributed Controller Architectures for Software Defined Networks

Heng Xu, **Baochun Li**, U. Toronto

Despite the merits of a logically centralized control plane in software-defined networks, it has the requirement to be implemented by physically distributed controllers to allow scaling up for larger networks. Such a problem of scalability becomes an even more pressing concern with application-layer software-defined networks. Such a physically distributed state naturally needs a way to keep the control planes working in unity, rather than just a number of independent networks. There are two broadly categorized models to synchronizing the physically distributed control plane, strong consistency and eventual consistency. In this poster, we show our recent work on an in-depth comparison study between these two models of consistency, with respect to a variety of performance and quality metrics.

Consistency Models in Distributed Controller Architectures for Software Defined Networks

Heng Xu, Baochun Li
Department of Electrical and Computer Engineering, Faculty of Applied Science and Engineering, University of Toronto

Introduction

Despite the merits of a logically centralized control plane in software-defined networks, it needs to be implemented by physically distributed controllers in order to scale up to a larger network. Such a problem of scalability becomes an even more pressing concern with application-layer software-defined networks. Such a physically distributed state naturally needs a way to keep the control plane to work in unity, rather than just a number of independent networks. There are two broadly categorized models to synchronizing the physically distributed control plane, strong consistency and eventual consistency. In this poster, we show our recent work on an in-depth comparison study between these two models of consistency, with respect to a variety of performance and quality metrics.

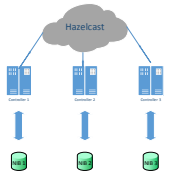
Motivation

Although a single instance of today's virtual machine (VM) can handle a large number of network events, but it still has its limitations. As a network grow, a single physically centralized controller will eventually met its scalability and reliability problems. Moreover, as the network grow, the physical placement of the control will be critical to the performance of the whole network. For example, as shown in the left, if we only have one physical instance of the controller, and two sites, no matter which site the controller is in, the switches located at the other site is going to incur large communication latency, and the cross site link may be congested. On the other hand, what if we have one instance of the controller for each site as whom on the right. This gives rise the idea of logically centralized, physically distributed network control plane.

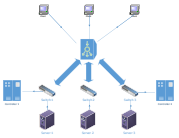
Prior Works

1) Onix: A Distributed Control Platform for Large-scale Production Networks, T. Koponen et al. in OSDI, 2010
2) HyperFlow: A Distributed Control Plane for OpenFlow, A. Tootoonchian and Y. Ganjali, in INM/WREN, 2010
3) Towards an Elastic Distributed SDN Controller (ElastiCon), Advait Dixit et al. in HotSDN, 2013
4) Logically Centralized? State Distribution Trade-offs in Software Defined Networks, D. Levin et al. in HotSDN, 2012

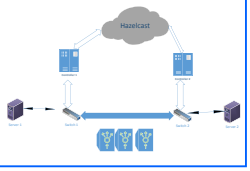
Core Concepts and Topology



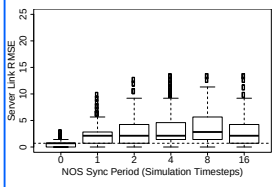
Let's understand the setup of the experiment. Similar to Onix, where each controller maintains a Network Information Base (NIB), to hold all the information about network entities. In our model, as shown on the left, each controller will maintain its separate NIB as well. Each phsyical instance of the controller will maintain its own NIB, which corresponds to its own view of the entire network. Since by default each controller will only be updated by the event in the segment of the network under its direct control, overtime the NIBs will evolve to different views. Therefore, there is a need to synchronize the NIBs contents, so that the views of the instances of controller will not be conflicting with each other.

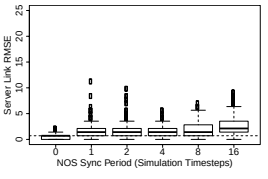


There are quite a number of SDN application to benchmark the results. There are quite a number of SDN applications we can use. For example, load balancers, middle boxes, firewalls. We have chosen the load balancer as our SDN application. As shown on the left, a number of clients send data to the backend servers. In-between the servers and clients, the load balancer will make an attempt to distribute the traffic evenly across all the paths, so that no link in the network is congested. The topology is here:



Performance Evaluation





We plotted network link imbalance measure boxplots. Under an exponential distributed workload with parameter alpha = 32. The eventual consistency case is shown on the left, while the strong consistency case is shown on the right. Both cases used 6 different synchronization periods between the controllers' NIBs and the distributed data store. We see the strong consistency model does better in terms of balancing link loads. This is just 1 out of many trails we performed.

FIGURE 1.2: *Consistency Models in Distributed Controller Architectures for Software Defined Networks from Heng Xu and* ***Baochun Li*** *at University of Toronto (Theme 1)*

1.3 Repairing Erasure Codes Cooperatively in Storage-Intensive Applications

Jun Li, **Baochun Li**, U. Toronto

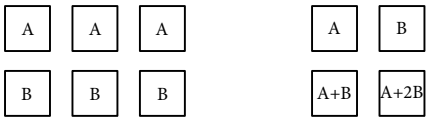
Storage-intensive applications store a substantial amount of data in a large number of commodity servers. Due to the large number and their commodity nature, servers are subject to failures. To maintain data integrity in spite of server failures, storage-intensive applications store redundant data, in a form of either replications or erasure codes or both. Since erasure codes can consume much less storage space than replications while maintaining the same level of failure tolerance, storage intensive applications have been increasingly deploying erasure codes (such as Reed-Solomon codes) to replace replications. However, when the storage-intensive application needs to repair missing data after failures, erasure codes also incur a significant amount of network traffic and disk I/O. Since storage-intensive applications are usually designed to tolerate multiple failures of servers, we can repair missing data of multiple failed servers in batches, rather than independently. In this manner, both network traffic and disk I/O can be saved with the corresponding erasure codes. In this paper, we present new erasure codes for storage intensive applications, called cooperative repairing (CR) codes that can repair missing data of multiple failed servers cooperatively while consuming the optimal network traffic. We show that the construction of CR codes is systematic and CR codes can be repaired exactly. Though CR codes incur more storage overhead than MDS codes like Reed-Solomon codes, we show that this overhead is marginal in storage-intensive applications. Our simulation results show that repairing multiple failures with CR codes can reduce network traffic by up to 20% and disk I/O of repairs by 75%.

Repairing Erasure Codes Cooperatively in Storage-Intensive Applications

Jun Li, Baochun Li
Department of Electrical and Computer Engineering, University of Toronto

Background: Tolerating Failures in Storage-Intensive Applications

- Storage-intensive applications, such as Hadoop and OpenStack, rely on redundant data to maintain data integrity in spite of server failures, in a form of either replications or erasure codes.

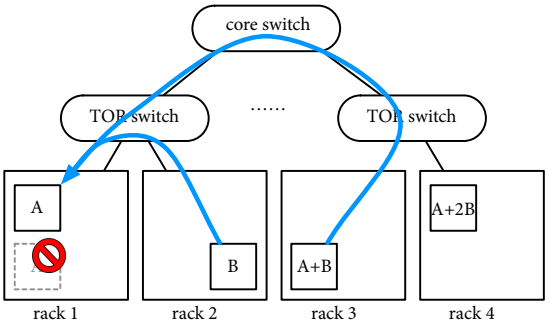


3-way replication (4, 2) erasure code

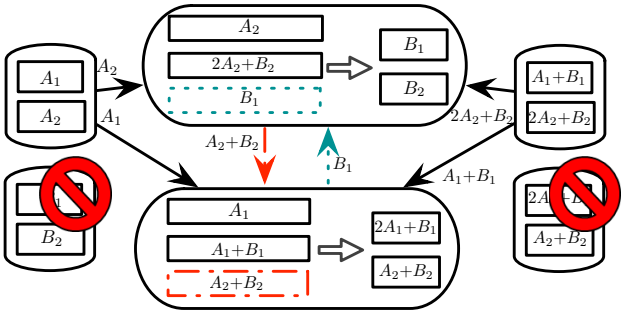
- Erasure codes consume much less storage space than replications, while maintaining the same level of failure tolerance. Hence, storage-intensive applications have been increasingly deploying erasure codes (such as Reed-Solomon codes) to replace replications.

Motivation

- However, erasure codes can also incur a significant overhead of network traffic (*esp.* over switches) and disk I/O, when the storage-intensive application needs to repair missing data after failures.



- We have known that network traffic and disk I/O can be saved if multiple failures are repaired in a cooperative way.



- Server failures are usually correlated, and thus there are many opportunities to repair multiple failures in batches, instead of one failure at a time.
- However, there has been no general construction of erasure codes that support to repair multiple failures in batches with the optimal network traffic.

Contributions

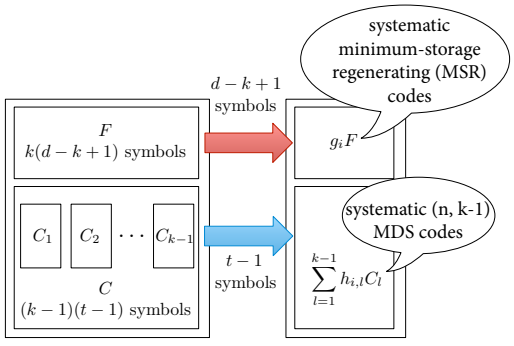
- We design new erasure codes for storage-intensive applications, called ***cooperative repairing*** (CR) codes.
- CR codes can repair missing data of multiple failed servers cooperatively while consuming the optimal network traffic with a marginal storage overhead.
- CR codes is systematic and CR codes can be repaired exactly.

Highlights of Code Design

- CR codes have four parameters: *(n, k, d, t)*:

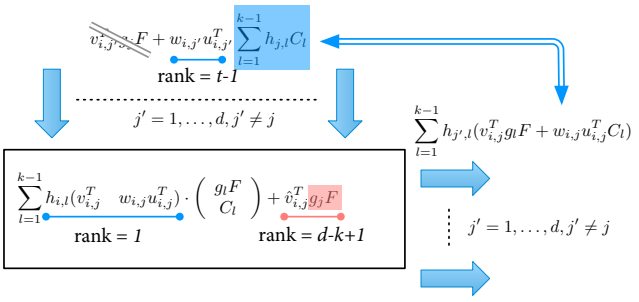
n	the number of blocks to be generated
k	the minimum number of blocks that can recover the original data
d	the number of existing blocks needed to repair the missing data
t	the number of failures repaired at the same time

- CR codes are constructed in two parts:

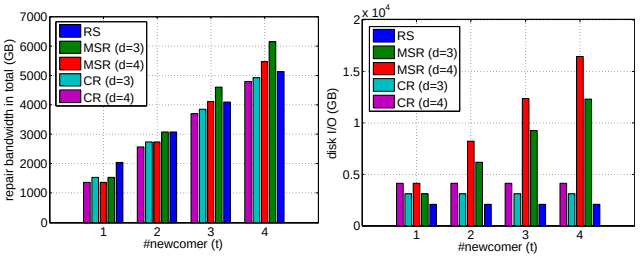


Any k blocks can recover the original data as both parts are MDS.

- CR codes can be repaired in a cooperative way:



Performance Evaluation



This work is funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP394424-10.

FIGURE 1.3: *Repairing Erasure Codes Cooperatively in Storage-Intensive Applications from Jun Li and* ***Baochun Li*** *at University of Toronto (Theme 1)*

1.4 Ready, Set, Go: Coalesced Offloading from Smart Applications to the Cloud

Liyao Xiang, **Baochun Li**, U. Toronto

With an abundance of computing resources, cloud-computing systems have been widely used to elastically offload the execution of computation-intensive smart applications on mobile devices. This leads to performance gains and better power efficiency. However, existing efforts have so far focused on one smart application only, and multiple applications are not coordinated when sending their offloading requests into the cloud. In this poster, we propose a new technique of coalesced offloading, which exploits the potential for multiple applications to coordinate their offloading requests with the objective of saving energy on mobile devices. Our intuition is that, by sending these requests in "bundles," the period of time that the network interface stays in the high-power state can be reduced.

We present two online algorithms, collectively referred to as Ready, Set, Go (RSG), that make near-optimal decisions on how offloading requests from multiple applications are to be best coalesced. We show, both analytically and experimentally, using actual smartphones, that RSG is able to achieve incremental energy savings whilst maintaining satisfactory performance.

Ready, Set, Go: Coalesced Offloading from Smartphones to the Cloud

Liyao Xiang and Baochun Li
Department of Electrical and Computer Engineering, University of Toronto

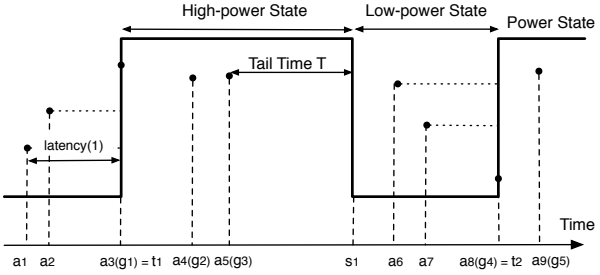
Motivation

When multiple applications send their offloading requests to the cloud independently without any coordination, the cellular or Wi-Fi network interface needs to be activated to transmit these requests, entering the high-power state at arbitrary times.

In this poster, we propose coalesced offloading, which exploits the potential for multiple applications to coordinate their offloading requests with the objective of saving additional energy on mobile devices. Coalesced offloading realizes the intuition that, by sending code offloading requests in "bundles," the period of time that the network interface stays in the high-power state can be reduced, thus saving additional energy.

The Coalesced Offloading Problem

Since bundling offloading requests may incur additional offloading delays, we formulate the problem as a joint optimization problem, with both the energy cost and the response time considered. The offloading requests arrival time sequence is $a_1, a_2, \ldots, a_9$, and the granting time sequence is $g_1, g_2, ..., g_5$. Let T be the duration of the tail time after the completion of transmission.



Since the actual energy cost is nearly linear to the duration that the network interface stays at the high-power state, we use that time duration to represent the energy cost. The joint optimization problem of coalesced offloading can be formulated as follows:

$$\min f_{\text{cost}} = \sum_j \min\{g_j - g_{j-1}, T\} + \alpha \sum_j \sum_{g_{j-1} \le a_i \le g_j} (g_j - a_i)$$

The first term represents the energy cost while the second term denotes the total latencies.

Ready, Set, Go: Algorithms

Optimal Offline Algorithm: The optimal offline algorithm, in which the arrival time sequence are given a priori, serves as the benchmark for us to design and evaluate our online algorithms. We use dynamic programming to obtain an optimal offline algorithm with a time complexity of O(n).

RSG Online Algorithm: Our algorithm A_θ is defined as a randomized algorithm that selects θ between 0 and 1 according to a probability density function $p(\theta) = e^\theta/(e-1)$. Let $R(t, t')$ be the number of requests that arrive between time t and t', and $g_1, g_2, ..., g_i, ...$ be the times at which requests are granted and transmitted. Algorithm A_θ grants the next request at g_{i+1} such that there exists a time τ_{i+1}, $g_i < \tau_{i+1} < g_{i+1}$, that satisfies

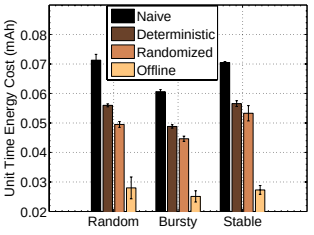
$$R(g_i, \tau_{i+1})(g_{i+1} - \tau_{i+1}) = (\theta/\alpha)S_i,$$

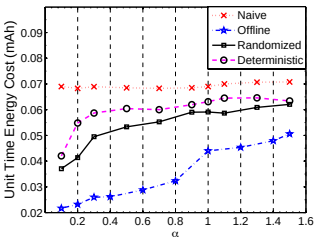
where S_i is essentially the amount of energy cost increment due to the additional transmission. The intuition is given the previous transmission occurring at time g_i, the additional transmission happens at τ_{i+1} will reduce the latency cost by the increment of energy cost.

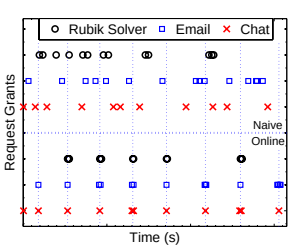
Performance Analysis: We proved the Deterministic Online Algorithm A_1 is 2-competitive, and the competitive ratio between the expected cost incurred by the Randomized Online Algorithm A_θ and the optimal cost is e/(e − 1), both of which are proven optimal in respective cases.

Performance Evaluation

In real-world experiments, we emulate different types of offloading requests generated from applications and their transfers to the cloud. We also look into how the energy costs vary with the tradeoff configuration parameter α. Further, we record the network traffic of three typical mobile applications offloading to the cloud. Our experiments have revealed that by performing the RSG algorithm with our real-world traces, the energy consumption is reduced by 20.71%.







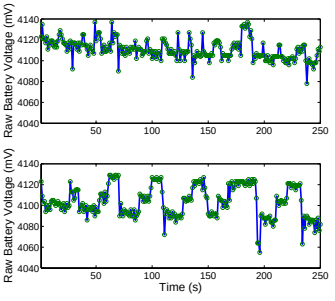


FIGURE 1.4: *Ready, Set, Go: Coalesced Offloading from Smart Applications to the Cloud from Liyao Xiang and* ***Baochun Li*** *at University of Toronto (Theme 1)*

1.5 Blossom: Content Distribution across Multiple Geographic Locations

Yangyang Li, **Baochun Li**, U. Toronto

Cloud service providers are building out geographically distributed networks of datacenters around the world. It is customary for cloud service providers to distribute their data replicas at multiple geographic locations to mitigate user latency and to increase service availability. In this poster, we treat the content distributed from one datacenter to multiple datacenters as a multicast session. We investigate the problem of maximizing the capacity utilization of inter-datacenter networks while maintaining fairness among multiple multicast sessions. A bandwidth allocation algorithm based on max-min fairness is developed, named Blossom. Blossom leverages a fully polynomial time approximation scheme to accelerate the bandwidth allocation, while achieving an approximation that is (1-epsilon)-optimal. Through trace-driven simulation, we show that our approach is substantially more efficient than prior work.

Blossom: Content Distribution across Multiple Geographic Locations

Yangyang Li, Baochun Li
Department of Electrical and Computer Engineering, University of Toronto

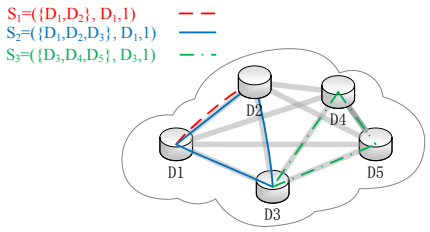
Introduction

Nowadays, it becomes a common practice for cloud service providers to build geographically distributed datacenters worldwide. In order to mitigate user latency and to increase service availability, it is necessary to distribute contents to multiple physical locations closer to users. With the proliferation of bandwidth-intensive applications, such as online video and file sharing, the scale of content distribution among these datacenters grows exponentially. The management of inter-datacenter traffic has drawn much attention in academia.

In this poster, we view that massive content distribution among datacenters as a bandwidth allocation problem for multiple multicast sessions. Rather than using the standard LP solvers, we are interested in developing an approximation algorithm that can solve the problem in polynomial time while achieving (1-epsilon)-optimality. In particular, we develop an algorithm called *Blossom*, which allocates bandwidth to multiple multicast sessions in a weighted max-min fair way. Our trace-driven simulation shows that *Blossom* outperforms existing multi-concurrent multicast solutions.

Motivation

We first use a toy example to illustrate the motivation for our idea.



➢***Session 1:*** Datacenter 1 to Datacenter 2, demand 1 unit.
➢***Session 2:*** Datacenter 1 to Datacenter 2 and 3, demand 1 unit.
➢***Session 3:*** Datacenter 3 to Datacenter 4 and 5, demand 1 unit.

Achieving optimized capacity utilization while maintaining proportional fairness among multicast sessions will result in a bandwidth allocation as (1, 1, 1). However, the bandwidth which is allocated to *Session 3* can be as high as 1.5 units without decreasing the allocation of a session with smaller or equal rates. This motivates us to choose ***max-min fairness*** as the criterion of bandwidth allocation for multiple multicast sessions.

Achieving weighted max-min fairness

According to the definition of max-min fairness, we have the following formulation for the max-min fair multi-tree multicast problem.

$$\textbf{P2}: lexmax_{x\in F} \quad \langle \boldsymbol{\lambda} \rangle$$
$$subject\ \ to \quad f_i(x) \geq \lambda_i \cdot dem_i, \forall i$$
$$\sum_{i=1}^{n} \sum_{j=1}^{|T_i|} x_i^j \cdot \delta_e(t_i^j) \leq c_e, \forall e \in \mathcal{E}$$
$$\lambda_i \geq 0, x_i^j \geq 0, \forall i, \forall j$$

Traditionally, based on the linear programming formulation, the problem can be solved by using a standard LP solver. However, as the number of multicast trees can be very large, finding the exact solutions by standard LP solvers can be slow and expensive. Instead, we developed a fully polynomial time approximation scheme (FPTAS). Below are highlights of our algorithm.

➢***Theorem 1.*** The running time of *Blossom* is

$$O(\epsilon^{-2} log|\mathcal{E}|(2n^2 logn|\mathcal{E}||\mathcal{V}| + n|\mathcal{E}|^2|\mathcal{V}|))$$

➢***Theorem 2.*** The resulted satisfaction ratio vector $\langle \boldsymbol{\lambda}^* \rangle$ is indeed max-min fair.

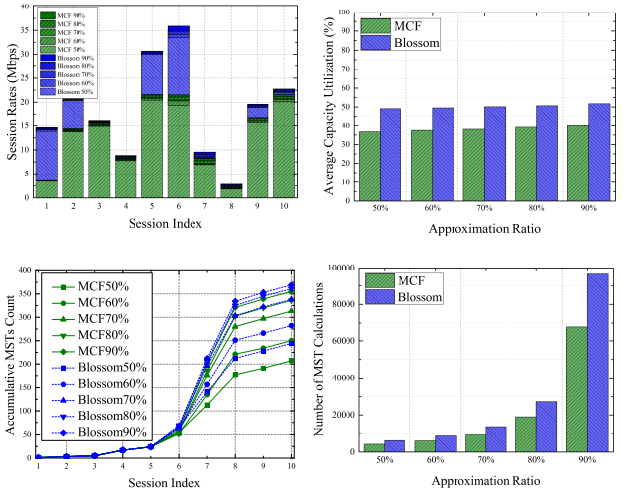
Performance Evaluation

Simulation settings:
➢Available bandwidth trace from 7 Amazon EC2 datacenters
➢10 multicast sessions are randomly created
➢Session demand range: [1, 10] Mbps
➢The number of datacenters involved in each session: [2, 7]

AVAILABLE BANDWIDTH (MBPS) FOR MICRO INSTANCES BETWEEN EC2 DATACENTERS. VA, CA, EU, SG, BR, OR AND JP CORRESPOND TO VIRGINIA, CALIFORNIA, IRELAND, SINGAPORE, BRAZIL, OREGON AND TOKYO, RESPECTIVELY

	VA	CA	EU	SG	BR	OR	JP
VA		24	35.4	16	26	19.1	18.2
CA	31.1		24.3	31.4	19	69.7	43.7
EU	110	9.65		21.7	13.8	14.6	7.13
SG	11.4	27.9	16		9.42	15	20.8
BR	29	15.4	16.8	13.8		18.4	4.24
OR	25.4	85.4	25.5	11.8	13.1		31.4
JP	31	35.9	16.9	12.4	3.85	54.4	

We evaluate the efficiency of *Blossom* compared to the multiple concurrent flow (*MCF*) algorithm which is using a proportional fairness criterion.



Conclusion

Our main contribution is an approximation algorithm that achieves max-min fair bandwidth allocation for competing multicast sessions in polynomial time. We carried out trace-driven simulation to evaluate the performance of *Blossom*, and showed that the algorithm can achieve high capacity utilization while maintaining inter-session fairness.

FIGURE 1.5: *Blossom: Content Distribution across Multiple Geographic Locations from Yangyang Li and* ***Baochun Li*** *at University of Toronto (Theme 1)*

1.6 Towards an Architecture for Mobile Social Video-Sharing Apps

Hu Zhang, Diego Serrano, **Eleni Stroulia**, U. Alberta

The typical architecture for web-based applications today involves a RBDMS back end, a middle tier implemented in a variety of programming languages and increasingly in the service-oriented style, and a web server listening for client requests and routing them to the appropriate service(s). Increasingly, these applications come with mobile clients that enable users to share and exchange multimedia content and access location- and time-aware services.

In this poster, we describe an architecture for developing such types of systems, on the SAVI cloud. The architecture includes components for (a) real-time social sharing of multimedia content; (b) data storage in cloud-based repositories, appropriately configured for different media types; (c) massively parallel processing for feature extraction; and (c) analytics services on the cloud. This work is motivated by the HSaT project, in the context of which we will be developing a system to support the caregivers of people with developmental disabilities.

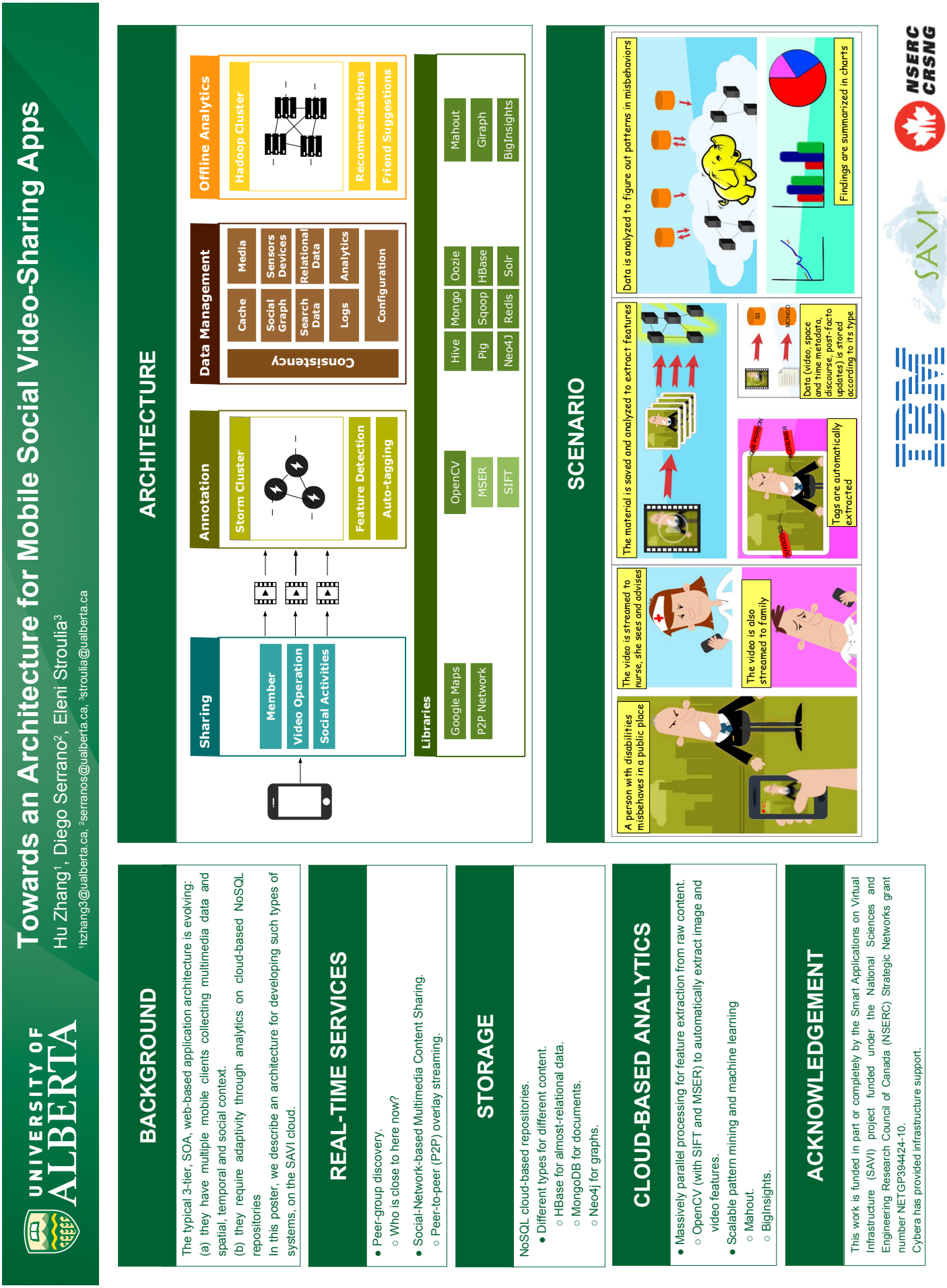


FIGURE 1.6: *Towards an Architecture for Mobile Social Video-Sharing Apps from Hu Zhang, Diego Serrano,* ***Eleni Stroulia*** *at University of Alberta (Theme 1)*

1.7 A Case Study on Multimedia Gaming Applications in Clouds

Preston Rodrigues, Ahmad Ferdous Bin Alam, Fariba Taheri, **Roch Glitho**, U. Concordia

Multimedia gaming applications are becoming ubiquitous. However, they still face several challenges (e.g. scalability, elasticity, easy development and deployment) that can be tackled with cloud computing principles. Several cloud based frameworks for multimedia gaming applications were proposed in the recent past. Nevertheless the vision of scalable, elastic, easily developed and deployed multimedia gaming applications remains quite elusive.

This case study focuses on multimedia multiparty games. An architecture that relies on fine grained gaming substrates is proposed. The substrates include multimedia conferencing substrates. They are sharable and can be assembled on the fly to develop and deploy new multimedia gaming applications. We have built an early prototype using Openstack at the IaaS layer, and Cloud Foundry at the PaaS layer. We also show how the proposed architecture can be deployed on SAVI test bed.

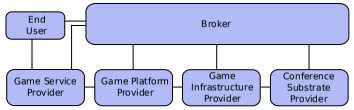
A Case Study for Multi-party Game Provisioning in the Cloud

Preston Rodrigues, Ahmad Ferdous Bin Alam, Fariba Taheri, Roch Glitho
{prestonr, a_binala, fa_tahe, glitho}@encs.concordia.ca

Motivation

Cloud computing has opened doors for providers (Software, Infrastructure, Platform) to package their offering as a *service*. Gaming, a multi-billion dollar industry, is expected to capture 60% of software market share by 2017. However, it still faces several challenges (e.g. scalability, elasticity, easy development and deployment) that can be tackled with cloud computing principles.

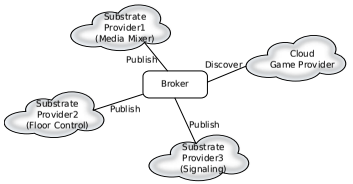
Proposed Approach



Business Model [1]

Game Semantics : Antakshari: a multi-party multimedia game [2]

- Game Moderator creates game and multiple teams.
- Game Participants join a team.
- Game Moderator selects one team at a time to sing.
- Time bound - point based - round robin.

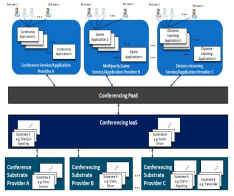


Third-party Substrates

Design Objective

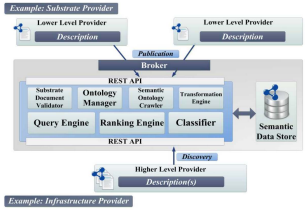
- A New Gaming Architecture based on fine grained building components (substrates) to scale in an elastic way.
- To develop and deploy games in SAVI testbed to prove the concept.

Architectural Components for multi-party Conferencing



Cloud Conferencing [3]

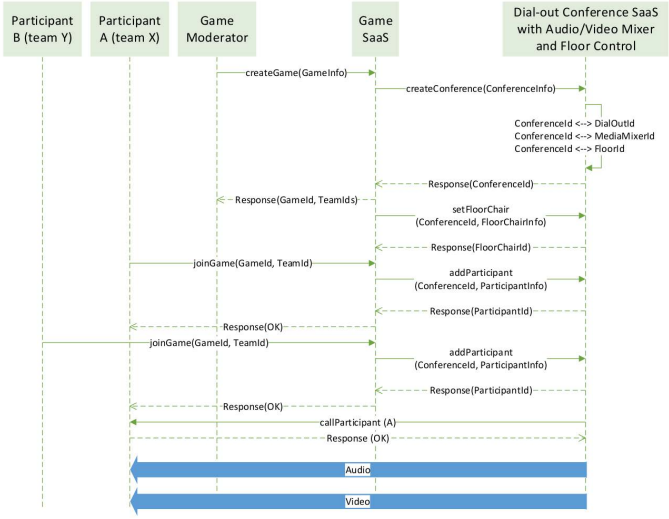
- Infrastructure based on fine grained substrates.
- Provides functionality for third-party substrate management.



Broker Architecture [4]

- Semantic Publication & Discovery.
- Provides RESTful APIs.

Game Flow



References

[1] Roch H Glitho. "Cloud-based multimedia conferencing: Business model, research agenda, state-of-the-art". In: (2011). C4E, pp. 226–230.

[2] Wikipedia. *Antakshari*. http://en.wikipedia.org/wiki/Antakshari.

[3] Flora Taheri et al. "A Cloud Infrastructure for Scalable and Elastic Multimedia Conferencing Applications". In: (2014). submitted to (IEEE-CNSM).

[4] Jerry George et al. "A Substrate Description Framework and Semantic Repository for Publication and Discovery in Cloud Based Conferencing." In: (2013). CSWS, pp. 41–44.

Future Work

- A PaaS that will enable developers to compose multi-party games using third-party substrate.
- Development of Resource Allocation Algorithms of the IaaS layer.

Acknowledgment

NSERC CRSNG — SAVI (NETGP394424-10)

Fonds de recherche sur la nature et les technologies Québec — FQRNT - nouveau chercheur

FIGURE 1.7: *A Case Study on Multimedia Gaming Applications in Clouds from Preston Rodrigues, Ahmad Ferdous Bin Alam, Fariba Taheri,* ***Roch Glitho*** *at Concordia University (Theme 1)*

1.8 Smart Applications with Green Deployments

Andreas Bergen, **Hausi A.Müller**, U. Victoria

A significant body of research exists to measure, monitor and control the resource utilization of virtual machines within data centers. However, within SAVI, we propose to investigate and monitor the energy consumption of smart applications at runtime. Energy consumption profiles are created by taking measurements at the server rack's power distribution unit (PDU) instead of using server internal hardware components. The resulting energy profiles of these virtual machines, hosting applications of users from around the globe, provide an improved understanding of "greenness" of the application within the context of the new "Green" initiatives. This allows us to make energy usage predictions as well as deployment decisions within SAVI.

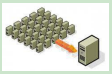
We propose to measure the energy consumption of various applications running within the SAVI network to build energy profiles and models of these applications. Since the abundance of applications in existence makes it unfeasible to profile every single one, we separate applications into categories depending on their resource usage: CPU intensive, memory intensive, intensive, disk I/O intensive. These profiles can then be used to determine the most energy friendly or "green" location to deploy smart applications within the virtual infrastructure of the SAVI Testbed.

Smart Applications with Green Deployments

Andreas Bergen , Hausi Müller
(andib@uvic.ca)
University of Victoria

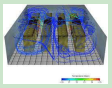
Related Work

VM consolidation [Beloglazov2010].



Migration of VMs to shutdown racks and server islands in data center [Mazzucco2012] .

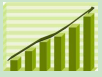
Other approaches: Improved cooling, frequency & voltage scaling, global energy perspectives, modelling, etc.



Approach

Profiling Energy Consumption of Software applications.

Migrate Applications depending on utility measurement: Performance, Time, Energy Consumption


vs

vs


Problem

2% of global energy consumption used for data centres.

Workload is not homogenous.




Under-utilized Servers.

Can we determine the dominant hardware resource a software application is using, based only on the energy profile?

Can we predict the energy consumption profile based on the resources a software system is/will be using?

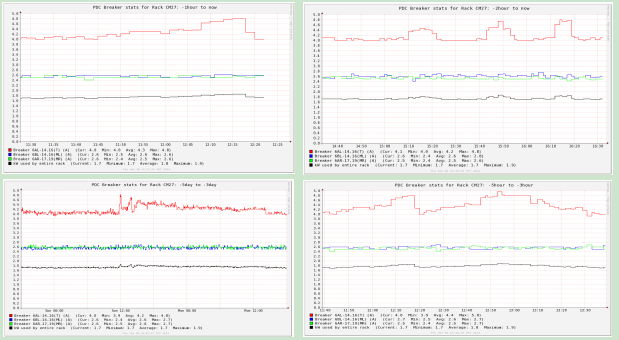
Can this information be used for smart adaptive scheduling and resource allocation models in data centres?

Evaluation

Memory Intensive applications: Energy consumption depends on memory usage pattern not on size used.

CPU: More resources do not result in improved performance, while more resources can result in energy savings.

Noticeable: Step functions clearly distinguishable and measurable



Future Work

Further Profiling.

Account for different hardware.

Account for different software applications.

Create models at runtime.

Dynamic scheduling decisions based on energy consumption of software A on hardware Z vs. software A on hardware Y.

FIGURE 1.8: *Smart Applications with Green Deployments from Andreas Bergen and* ***Hausi A.Müller*** *at University of Victoria (Theme 1)*

1.9 Slicing Applications in the SAVI Pie

Przemek Lach, Ron Desmarais, Thiago Lima, **Hausi A.Müller**, U. Victoria

Modern applications are designed with performance, redundancy, scalability, and cost saving in mind. The arrival of the cloud and multi-cloud models has offered better tools for implementing these characteristics but at the cost to the developer: building a multi-cloud application is complex. One such modern application is Yakkit Chat. Yakkit Chat is location based chat application that can take advantage of the multi-cloud architecture. Our demonstration shows how SAVI eases the burden of development for a multi-cloud model and how that translates to cost savings for the developer and an improved experience for the user.

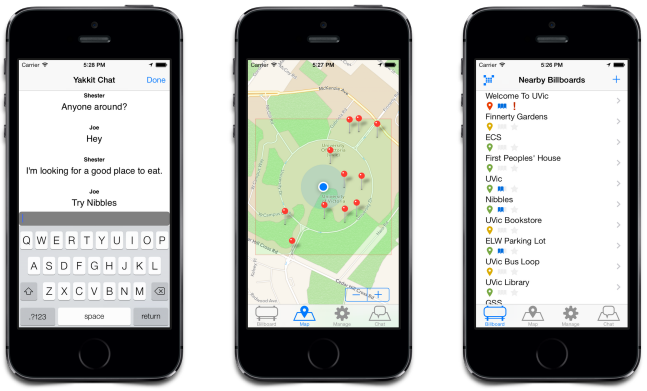
SLICING APPLICATIONS IN THE SAVI PIE

Przemek Lach, Ron Desmarais, Thiago Lima, Hausi Müller

University of Victoria

Yakkit is a framework for building context aware applications.

Yakkit Chat is an application, built using the Yakkit Framework, that allows you to communicate instantly with those around you. One of the goals of Yakkit Chat is to deliver a social experience that is similar to a face to face conversation. This goal necessitates a certain level of responsiveness from the application that is susceptible to latency. Using SAVI, Yakkit Chat has the ability to make better run-time decisions that help mitigate latency.



Experiment Setup

The setup is composed of four servers: two on the West Coast and two on the East Coast. The Victoria Edge and Carleton Edge servers are part of the SAVI infrastructure and are running the Yakkit Chat server. The Amazon EC2 Oregon and Amazon EC2 Virginia servers are part of the Amazon EC2 cloud and they are running bots that simulate users engaged in conversations using the Yakkit Chat app.

Figure 1: Connection Setup for Potentially Low Latency Scenario.

Oregon Users (West Coast)

Since Carleton is further away form Oregon than it is from Victoria, Oregon users should see lower latencies using the Victoria Edge rather than the Carleton Edge. We observed that on average it took a message 0.2178 seconds to be delivered via the Victoria Edge and 0.2410 seconds through the Carleton Edge. These results indicate that at that point in time users would observe a better experience using the Victoria Edge.

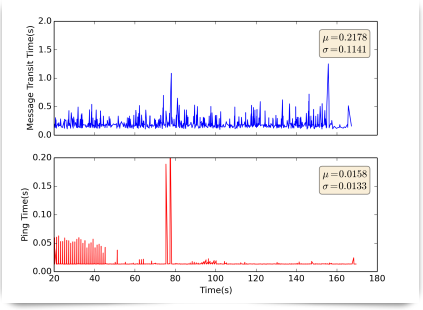


Figure 2: Oregon to Victoria with ten bots and five 150 second runs.

Virginia Users (East Coast)

Since Victoria is further away form Virginia than it is from Carleton, Virginia users should see lower latencies using the Carleton Edge rather than the Victoria Edge. We observed that on average it took a message 0.1843 seconds to be delivered via the Carleton Edge and 0.2590 seconds through the Victoria Edge. These results indicate that at that point in time users would observe a better experience using the Carleton Edge.

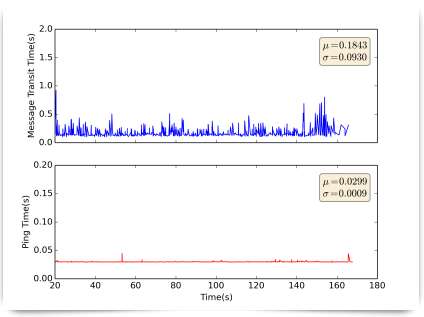


Figure 3: Virginia to Carleton with ten bots and five 150 second runs.

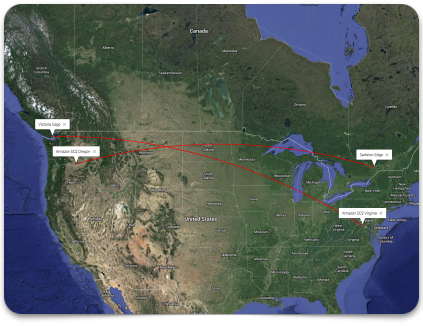


Figure 4: Connection Setup for Potentially High Latency Scenario.

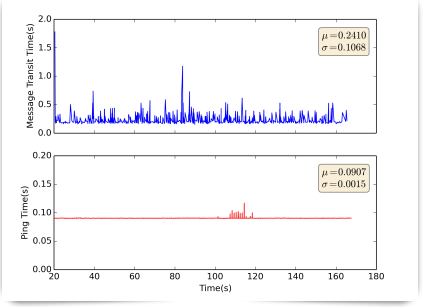


Figure 5: Oregon to Carleton with ten bots and five 150 second runs.

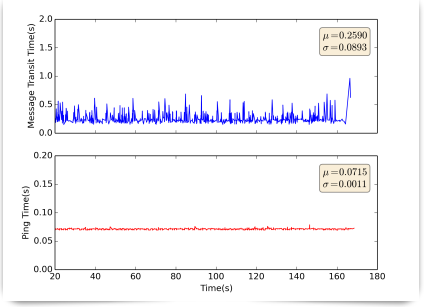


Figure 6: Virginia to Victoria with ten bots and five 150 second runs.

SAVI 2014

FIGURE 1.9: *Slicing Applications in the SAVI Pie from Przemek Lach, Ron Desmarais, Thiago Lima and* ***Hausi A.Müller*** *at University of Victoria (Theme 1)*

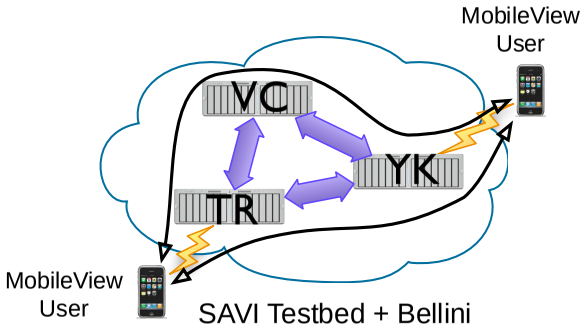
1.10 MobileView: Optimized Mobile Video Conferencing over the SAVI Testbed

Zimu Liu, **Baochun Li**, U. Toronto

With the proliferation of mobile devices with high-speed cellular networks and high-resolution cameras, there is a great potential that mobile video conferencing could greatly enhance the socialization and communication among users. In this project, we aim to provide a "smart" video conferencing service across geo-distributed SAVI edges, to offer the best possible service. As the cornerstone of our smart application, we first conduct thorough measurements on the SAVI infrastructure to study its inherent characteristics. Compared to traditional video conferencing, we discovered that the SAVI infrastructure could significantly boost the streaming throughput, while keeping both delay and delay jitter low. Based on our measurement insights on SAVI, we carefully design a suite of dedicated algorithms to stream mobile video flows through SAVI edge nodes, and our algorithms consider various critical factors, including streaming rate, latency, and operational costs. To validate our design for the smart mobile video conferencing, we have developed a video conferencing application on the iOS platform from the scratch, named MobileView. Our experiments reveal that the MobileView application is able to take full advantage of the SAVI testbed and offer a smooth and high-quality conferencing experience in mobile devices.

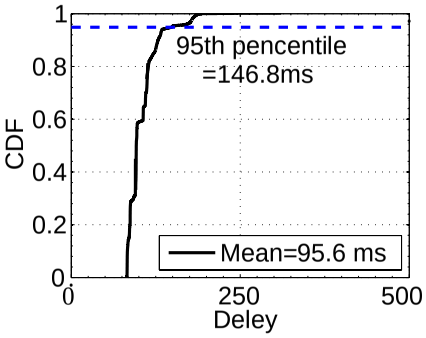
MobileView: Optimized Mobile Video Conferencing over the SAVI Testbed

Zimu Liu and Baochun Li
University of Toronto



Objectives:

- Ensure the expected latency
- Minimize the delay jitter
- Increase throughput
- Reduce costs



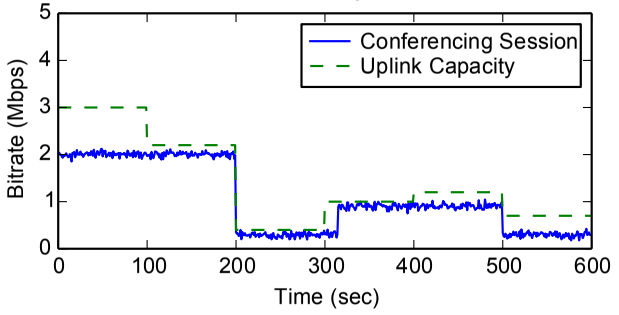


FIGURE 1.10: *MobileView: Optimized Mobile Video Conferencing over the SAVI Testbed from Zimu Liu and* ***Baochun Li*** *at University of Toronto (Theme 1)*

1.11 GLINT – A Horizon based Image Distribution System

Ronald Desmarais, **Colin Leavett-Brown**, U. Victoria

Glint is a cloud image distribution service built using the django framework. Glint currently supports Openstack clouds. Glint extends Horizon to add image distribution functionality to Horizon's Image Management interface. Glint uses Openstack's Glance Image Management API to manage images on multiple Openstack clouds in a consolidated way. Glint offers two main services: the Site Management and User Credential Service, and the Image Distribution Service. The Site Management and User Credential Service enables users to identify other Openstack sites. It also allows users to store their credentials for sites, which Glint uses for authorization to modify that site's Glance repository. The Image Distribution Service uses Glance to identify all images across all sites and creates a simple table for the user to easily select which images they want to have on selected sites. Glint uses Glance to transfer images from source sites to destination sites using the Glance API.

HEP Group, UVic July 2, 2014

GLINT

R.Desmarais | C.Leavett-Brown | F.Berghaus | I.Gable | R.Sobie | A.Lam | C.Dremiel | R.Taylor | M.Paterson

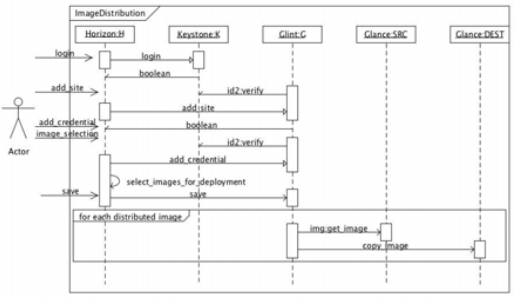
Objective

Provide an image replication service managed through Openstack's Horizon Interface, to aggregate image management over multiple cloud sites including Openstack, EC2 (Amazon) and GCE (Google).

Motivation

The popularity of cloud software has increased the number of cloud providers. For users to take advantage of these sites, they require credentials for access. This requires an administrative

burden to users in setting up their VM images across many sites. Such a process requires the user to login and push or remove their images on every site they have access to. Users that have access to many sites, such as HEP, be burdened by image management.



Openstack Based Design

We decided to heavily rely on the Openstack framework for the following:

1. To use Openstack's Glance service and API to manage image distribution.

2. To take advantage of Openstack's development and deployment architecture.

3. To use Openstack's keystone service to provide user authentication.

4. To use a secure mechanism to copy images.

Simple User Interface

1.User Adds Openstack Sites they have credentials on.

2. User Adds their Credentials for each site they added.

3.User selects/unselects sites they wish to push their images.

University of Victoria Supported by CANARIE Dept. of Physics and Astronomy

canarie

FIGURE 1.11: *GLINT – A Horizon based Image Distribution System from Ronald Desmarais,* ***Colin Leavett-Brown****, at University of Victoria (Theme 1)*

Page for Notes

Chapter 2

Theme 2 Extended Cloud Computing

Research Team

Marin Litoiu (Theme Lead), York University
John Chinneck, Carleton University
Kenneth Salem, University of Waterloo
Murray Woodside, Carleton University
Michael Smit, Post-Doctoral Fellow, York University

2.1 Task Assignment Across Clouds by Graph Partitioning

Ravneet Kaur, **John Chinneck** and **Murray Woodside**, Carleton U.

Task assignment in cloud computing normally ignores inter-task communications on the assumption that this is a minor effect, but communications latency can have a big impact in newer cloud architectures where the cloud consists of multiple computing centers of various sizes and the inter-cloud communication times are not negligible. We study the case of task partitioning between a main "core" cloud and a smaller "edge" cloud closer to the end user, where the edge-core communication time is not negligible. Transactions having heavy interaction with the user are best placed in the edge cloud, and those requiring heavy computation with less communication are best placed in the core. We propose iterative graph partitioning methods that assign tasks in the task graph to the edge or core by minimizing a cut related to communication volumes and processing capacity. Initial experimental results are promising. of the cloud.

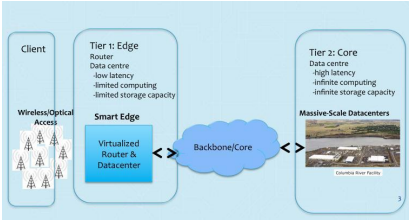
Task Assignment Across Clouds by Graph Partitioning

Ravneet Kaur, Dr. John W. Chinneck, Dr. C. Murray Woodside
System and Computer Engineering
Carleton University, Canada

SCOPE

Inter-task communications have a big impact in newer cloud architectures .Cloud consists of multiple computing centers of various sizes.
Task partitioning between a main "core" cloud and a smaller "edge" cloud closer to the end user, where the edge-core communication time is not negligible.
We propose iterative multilevel graph partitioning methods that assign tasks in the task graph to the edge or core by minimizing a cut related to communication volumes and processing capacity.

Extended cloud architecture

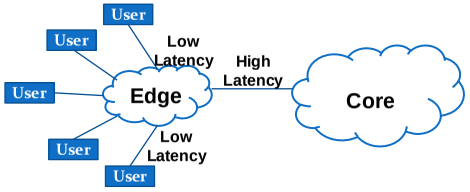


Why is low-latency needed?

- Characteristics of future application
  - massive amounts of data (e.g. video)
  - tight real-time response time constraints
- Examples of future-oriented applications
  - Multi-player interactive games
  - Battlefield situational awareness
  - Sensor integration
  - Disaster response
  - Video integration

How to deploy tasks?

- **Communications latency** can have a big impact
- **Varying size** multiple computing centers
- **Task partitioning** between "core" cloud & smaller "edge" cloud
- **Heavy interaction** transactions best placed at edge
- **Less communicating heavy computation** best placed at core



Objectives

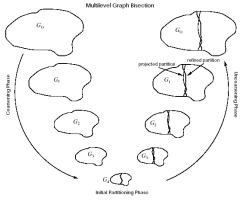
- Minimise power consumption
- Minimise communication latency (Inter-task communication across clouds)

Constraints

- Response time limit
- Host processer capacity
- Memory limit

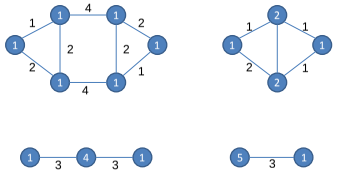
Multilevel Graph Partitioning

- 3 Phases
  - Coarsen
  - Partition
  - Uncoarsen & Refine



Coarsening

- Vertices are combined to give a smaller graph
- Beneficial to collapse heavy weighted edges
- Algorithms:
  - Heavy Edge Matching (HEM),
  - Heuristic task assignment
  - Random matching,
  - Modified Heavy Edge Matching (MHEM)



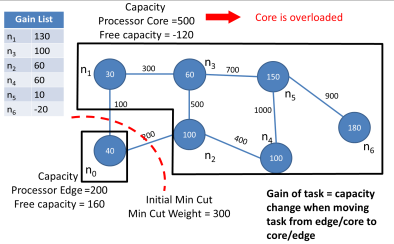
Uncoarsen & Refine Partition

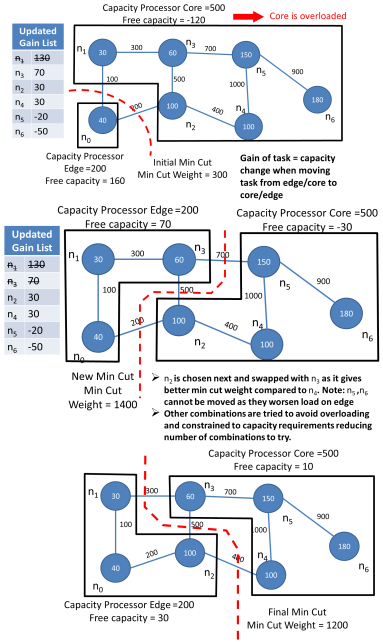
- Uncoarsening: the partitioning of the coarser graph Gm is projected back to the original graph by going through the graphs G_{m-1}, G_{m-2},…, G_1.
- Refine partition: a local partitioning refinement algorithm required like Kernighan-Lin (KL) algorithm, Simulated Annealing (SA), Hybrid KLSA
- Very fast phase and works well if coarsening done correctly

Proposed Algorithm

Compute initial partition via Stoer-Wagner Min Cut algorithm
Compute initial gains and free space of edge & core
Best partition = current partition
Repeat
  For overloaded partition
    Find task with highest gain
    If Free(not_overloaded_partition) - task_capacity > Free(overloaded_partition)
      Move this task
      Update gains and free space
    If current partition < Best partition and both edge/core not overloaded
      Best partition = current partition
If no partition overloaded
  Repeat
    Perform task swaps and update Best partition if current partition < Best partition and both edge/core not overloaded

Example: Algorithm





- n_2 is chosen next and swapped with n_1 as it gives better min cut weight compared to n_0. Note: n_5,n_6 cannot be moved as they worsen load on edge
- Other combinations are tried to avoid overloading and constrained to capacity requirements reducing number of combinations to try.

Contributions

+ End-user response times limits satisfied for various combinations of edge-cloud application deployments
+ Faster algorithm
+ Stoer-Wagner Min-cut minimises communication, execution cost in initial cut and capacity constraints satisfied

Limitations and Future Work

- Single edge-sized and core-sized cloud
- KL/FM can get stuck in local minima
- Refinement and testing of task modifying function
- Vigorous testing on data set
- Extension to multi-cloud
- Power consumption issue to be addressed
- Memory Limit to be included

This work is funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP394424-10

FIGURE 2.1: *Task Assignment Across Clouds by Graph Partitioning from Ravneet Kaur,* ***John Chinneck, Murray Woodside*** *at Carleton University (Theme 2)*

2.2 Simulation of Future Application-Aware Multi-Clouds

Derek Hawker, **John Chinneck** and **Murray Woodside**, Carleton U.

We have extensively modified the DCSim cloud simulator for use in examining the impact of latency on the response times of applications running on multi-cloud architectures. The main new features are the integration of the analytic Layered Queuing Network Solver (LQNS), and an expanded application model based on Layered Queuing Networks (LQN).

Together these provide response times based on an LQN application model that reflects how the underlying VM deployment creates latencies between tasks. Also new is support for the creation of a variety of multi-cloud architectures.

Simulation of Future Application-Aware Multi-Clouds

Derek Hawker, John Chinneck, Murray Woodside
Carleton University, Ottawa, ON

Abstract

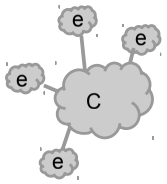
Our simulator can be used to simulate applications running in a multiple datacenter cloud like the SAVI testbed. It enables the development and testing of VM allocation policies in a controlled environment. In this environment we are able to assess the impact of factors, such as latency, on application performance.

Motivation/Future Cloud Networks

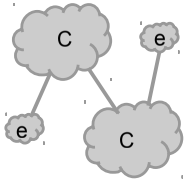
Our interest in developing this simulator is the study of future applications. These applications have numerous characteristics that make them difficult implement on current available hardware. Future applications will have high bandwidth requirements, contain strict real-time requirements and exhibit highly dynamic behaviour.

To power these future applications we will need new cloud models. One of the keys to meeting satisfying latency requirements, will be the location of datacenters and VM placement within the cloud. From the perspective of the cloud operator, a level of application-awareness is necessary for optimal VM placement.

A simulator is beneficial because we are able to simulate the characteristics of future applications. These applications can be deployed on different multi-cloud architectures against different kinds of VM allocation policies. All in a controlled and repeatable environment.



SAVI testbed
one core, multiple edges



Multiple cores

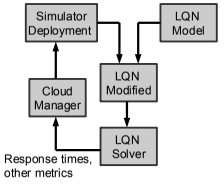
Simulator Features

Began with existing simulator (DCSim)
- Customizable VM Allocation policies
- IAAS-perspective

Multiple Improvements
- Integration of analytic solver called LQNS (Layered Queuing Network Solver)
  - Improved quality of response time measurements
  - Layered queuing network delays
- Response time analysis includes latency delays between related tasks
  - Within rack
  - Within cluster
  - Within datacenter
  - Between datacenters
- Expanded datacenter model to manage multiple datacenters
- Improved application model based on Layered Queuing Networks

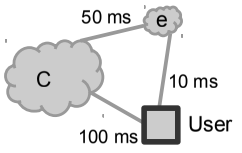
Updated Simulator Event Loop

New event loop calls out to LQN solver. A custom LQN is constructed for the current deployment of VMs based on a provided LQN model of the application.



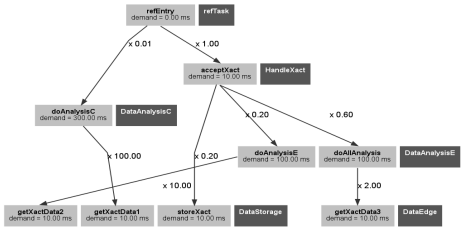
Example

HCAT is a healthcare management application. Several layers of communication between tasks. Some heavy processing tasks best suited to the core. Heavy communication between some layers makes for better candidates closer to the user.



Simulated Cloud

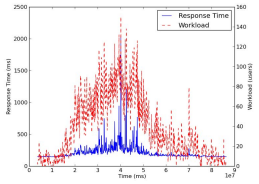
Parameter	Value
# Edge Host	2
# Core Host	3
Host Type	(1)core, 2.5 Ghz, 16GB, 1Gbps

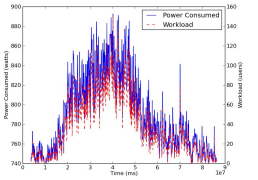


Simulated Application

Results

After simulating 24 hours





Comparison Of Simulators

- Queuing from bottleneck tasks not represented in original DCSim
- Improved simulator detects response times that are unstable and will not converge.

	Original DCSim	Improved Simulator
Response time (ms) (min, avg, max)	(109,124,230)	(147, 209, 2035)
Total utilization (%)	9.63	10.11
Power usage (Ws) (min, avg, max)	(1393, 1430,1538)	(1541,1589, 1692)

Future Work

- Validation of Results
  - Completing work on a Layered System Generator (LSG) that deploys to the SAVI testbed.
  - Simulator results will be compared with testbed measurements.
- Implementation of VM allocation policies.

References

Chinneck, J., Litiu, M., & Woodside, M. (2014, March). Real-time multi-cloud management needs application awareness. In Proceedings of the 5th ACM/SPEC international conference on Performance engineering (pp. 293-296). ACM.
Tighe, M., Keller, G., Bauer, M., & Lutfiyya, H. (2012, October). DCSim: A data centre simulation tool for evaluating dynamic virtualized resource management. In Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualization management (svm) (pp. 385-392). IEEE.
Faisal, A., Petriu, D., & Woodside, M. (2013, November). Network latency impact on performance of software deployed across multiple clouds. In Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research (pp. 216-229). IBM Corp.

Acknowledgements

This work is funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP394424-10

NSERC CRSNG

SAVI

FIGURE 2.2: *Simulation of Future Application-Aware Multi-Clouds from Derek Hawker,* ***John Chinneck, Murray Woodside*** *at Carleton University (Theme 2)*

2.3 Relational Edge Caching for Edge-Aware Web Applications

Hemant Saxena and **Kenneth Salem**, U. Waterloo

Web application latencies can be reduced by migrating all or parts of the application to the network edge, closer to end users. However, web applications normally depend on a back-end database, and moving the application to the edge without moving the database is of little value.

To address this problem, we present preliminary work towards an edge-aware dynamic data replication architecture for relational database systems. Our architecture is designed to support applications that rely on substantial amounts of end-user-generated content. To do so, it must allow database updates as well as queries, to be handled at the edge.

Relational Edge Caching for Edge-Aware Web Applications

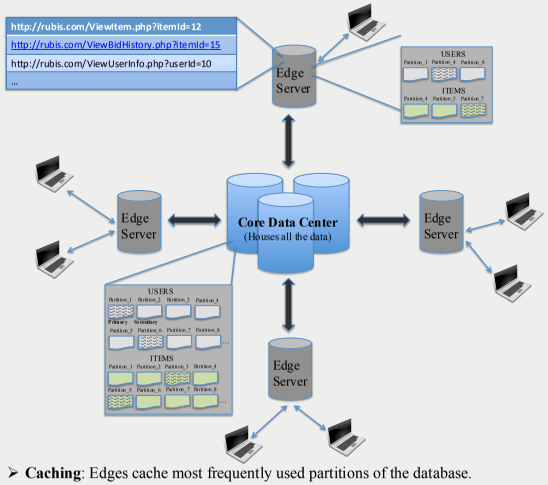
Hemant Saxena, Kenneth Salem

UNIVERSITY OF WATERLOO

Introduction

- With the global expansion of web applications like Ebay, Craigslist and Groupon, data centers have become more distant from the users.
- Content Delivery Networks (CDNs) are servers sitting close to users and caching static web content.
- **The Idea** of this project is to enable the network edge to do more than cache static content. In particular, we want to allow operations that depend on dynamic content to be performed at the edge.

Architecture

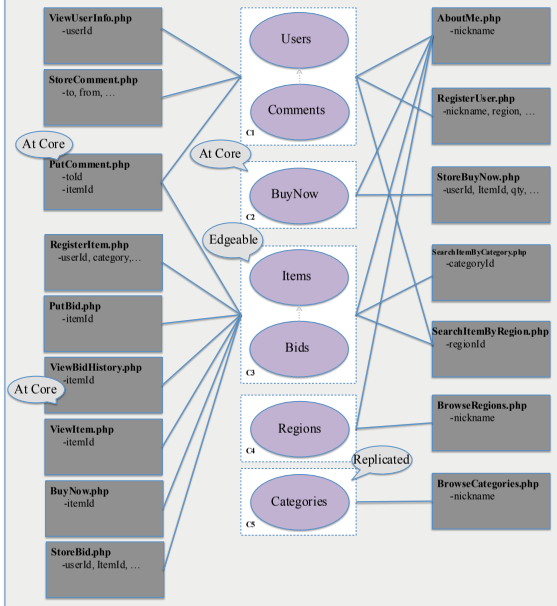


- **Caching**: Edges cache most frequently used partitions of the database.
- **Primary site**: Each partition has a primary site, which is responsible for handling updates.
- **Updates**: Primary site asynchronously sends updates to secondary sites.
- **Transaction**: Distributed transactions are not allowed.
- **Metadata**: Each edge maintains metadata describing the partitions cached there. These metadata are used to determine whether an application operation can be performed at that edge.

Example

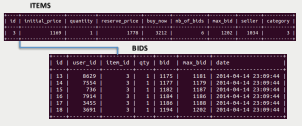
How do we decide placement of tables between Edge and Core?

Example – RUBiS is an Ebay like web application, where user can view, buy, bid and comment on items.



Observations & Conclusions:

- **Single parameter** - Most of the PHP operations use single parameter in the predicate of the underlying SQL query.
  - *Therefore, just by looking at the parameter value in the PHP request, the availability of the required data at the Edge can be checked.*
- **Clustering** - Table pairs like Users and Comments, Items and Bids, are related via foreign keys.
  - *These tables should be put together in one cluster, as operations access them via join queries.*



- **Partitioning** – The clusters having tables related via foreign keys reflect hierarchical partitioning.
  - *Therefore, such clusters can be partitioned into chunks defined by the join queries over the tables.*
- **Static & Runtime policies** – Operation ViewBidHistory.php accesses cluster C3, therefore, it can be handled at the Edge and partitions of C3 will be moved to the edge at runtime. Operation StoreBuyNow.php is accessing a non-partitionable cluster and needs strong consistency, therefore, should go to the Core.
  - *Therefore, given the operation type, consistency requirements and partitioning information, we can decide the execution policy for operations and placement policy for clusters.*

Problem Definition

- **Aim** – to come up with an optimal placement of clusters between edges and core, so that the number of operations that can be performed at the edge is maximized.
- **Given**– the following information about each operation-cluster pair:
  - *Is the operation performing reads or updates on the cluster?*
  - *Is the cluster partitionable?*
  - *Does the operation need fresh data?*
- **Output**– assign one of the following label to each cluster:
  - **Replicated**: *the cluster is replicated at all the edge sites and at the core site..*
  - **At Core**: *the core will be the primary site and secondary will be present at the edges.*
  - **Edgeable**: *the cluster is partitioned, and the partition's primary site is moved to an edge on demand.*

Future work

- The problem of assigning labels based on the input constraint for each Operation-Cluster pair can be transformed into a SAT problem.
- Second part of the problem is to convert it to a optimization problem, like an Integer Linear Programming problem, to find out the best placement strategy.

Conclusion

- Global web applications still face a problem of high network latency for dynamic content, such as query results over database, the reason being that the database is hosted at a distant data center, and Edge networks only cache static web content.
- A Core and Edge architecture utilizes the already existing Edge infrastructure to provide database features at Edge level. This can result in low network latency for dynamic content as well.
- In an Edge infrastructure, the placement of database between Core and Edge becomes crucial for the system performance.
- We are working towards a methodology to optimally place the data in an Edge infrastructure that results into maximum Edge utilization by the web application.

Acknowledgement

This work is funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP394424-10.

NSERC CRSNG

FIGURE 2.3: *Relational Edge Caching for Edge-Aware Web Applications from Hemant Saxena,* **Kenneth Salem** *at University of Waterloo (Theme 2)*

2.4 Optimal Service Replica Placement via Predictive Model Control

Hamoun Ghanbari, Przemyslaw Pawluk, Cornel Barna and **Marin Litoiu**, York U.

We present a model and an algorithm for optimal service placement (OSP) of a set of N-tier software systems. The placement is subject to dynamic workload changes, Service Level Agreements (SLAs) and administrator preferences. The objective function consists of resource costs, trashing costs and SLAs' satisfaction. The optimization algorithm is predictive: its allocation or reallocation decisions are based not only on the current metrics but also on the predicted evolution of the system.

The solution of the optimization, in each step, is a set of some service replicas to be added or removed from the available hosts. These deployment changes are optimal with regards to overall objectives defined over time.

Optimal Service Replica Placement via Model Predictive Control

***Hamoun Ghanbari*[1], *Marin Litoiu*[1], *Przemyslaw Pawluk*[1], *Cornel Barna*[1]**

[1]*Dept. of Computer Science, York University, Toronto*

YORK UNIVERSITÉ UNIVERSITY

Introduction

We present a model and an algorithm for optimal service placement (OSP) of a set of N-tier software systems, subject to dynamic changes in the workload, Service Level Agreements (SLA), and administrator preferences. The objective function models the resources' cost, the service level agreements and the trashing cost. The optimization algorithm is predictive: its allocation or reallocation decisions are based not only on the current metrics but also on predicted evolution of the system. The solution of the optimization, in each step, is a set some service replicas to be added or removed from the available hosts. These deployment changes are optimal with regards to overall objectives defined over time.

Figure 1: *An example of placement decisions for a small service center using step based optimization, ignoring the future steps.*

Proposed Method

The deterministic optimal control program, solved at each step, providing an answer to the abstract stochastic planning problem.

given: $R^{SLA}_{c,j}$ for each c, j; $\theta_{s,h,0}$ for each s and h, Ω_h for each h, $t, J, r_{\text{resource}}, r_{\text{dep}}, r_{\text{SLA}}$

$$\underset{u_{t=\cdots,tJ-1}}{\text{minimize}}\ r_{\text{SLA}} \sum_{j=t}^{J}\sum_{c=1}^{C} pos(X^{\text{SLA}}_{c,j} - X_{c,j}) + r_{\text{resource}} \sum_{j=t}^{J}\sum_{h=1}^{H}\sum_{s=1}^{S}\sigma^*_{s,h}\sum_{c=1}^{C} X_{c,j} d_{c,s}\theta_{s,h,j} + r_{\text{dep}} \sum_{j=t}^{J}\sum_{h=1}^{H}\sum_{s=1}^{S} |u_{s,h,j}|$$

subject to:

$$X^{\text{SLA}}_{c,j} = N^{\text{pred}}_{c,j} / (Z^{\text{pred}}_{c,j} + R^{\text{SLA}}_{c,j}) \quad \text{for each } c, j$$

$$\theta_{s,h,j+1} = \theta_{s,h,j} + u_{s,h,j} \quad \text{for each } s, j$$

$$\sum_{h=1}^{H} \theta_{s,h,j} = 1 \quad \text{for each } s, j$$

$$U_{h,j} \le L\Omega_h\sigma_h \quad \text{for each } h, j$$

$$U_{h,j} = \sum_{c=1}^{C} X_{c,j} \sum_{s=1}^{S} d_{c,s}\theta_{s,h,j} \quad \text{for each } h, j$$

Nomenclature

Ω_h The multiplicity of host h
r_{dep} The trashing cost coefficient
r_{resource} The resource cost coefficient
r_{SLA} The SLA violation cost coefficient
$\sigma^*_{s,h}$ A host-service specific cost coefficient based on utilization.
σ_h The speed factor of host h
$\theta_{s,h,t}$ The portion of service s that is handled by host h at time step t through a placed replica and proper routing.
$U_{h,k}$ Utilization of a resource k of a host h
C The number of classes of customers in the workload
$d_{c,s}$ the total service demand made on service s, by a single request of class c.
H Number of hosts in a datacenter
K Number of different types of resources in a datacenter (i.e. the CPU, disk, and the network)
$N^{\text{pred}}_{c,j}$ Predicted number of users in class c at time j
$pos(x)$ Positive value of a real number or $max\{x, 0\}$.
$R^{SLA}_{c,j}$ SLO upper-bound on the response time of class c at time j.
S The number of services in the datacenter.
t Current time step index
$u_{s,h,j}$ Change in the placement of the service replica of the service s on host h made at time t.
$X_{c,j}$ Average system throughput for class c customers at time j
$X^{\text{SLA}}_{c,j}$ SLO upper bound on the throughput of class c at time j.
$Z^{\text{pred}}_{c,j}$ Predicted think time of class c at time j
Z_c Think time of class c

Experiment Set 1

We performed several experiments to test the response of the controller over time.

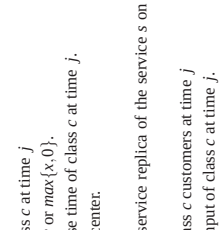


Figure 2: *The cumulative distribution function (CDF) of the number of relocations for the resource precedent (left) and trashing precedent (right) controller configurations.*

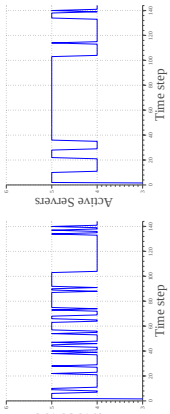


Figure 3: *The number of active servers over time for the resource precedent (left) and trashing precedent (right) controller configurations.*

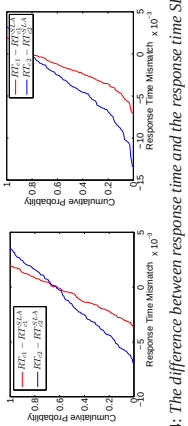


Figure 4: *The difference between response time and the response time SLA as a CDF for the resource precedent (left) and trashing precedent (right) controller configurations.*

Experiment Set 2

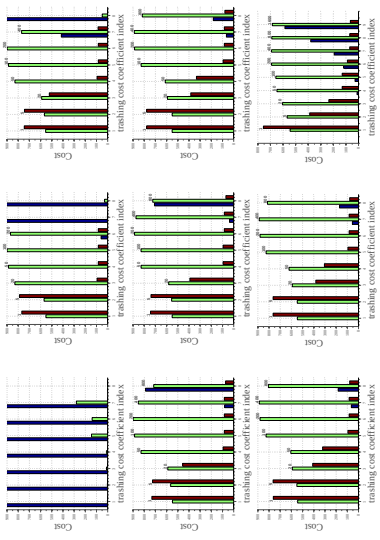


Figure 5: *the cost trade-off curves achieved by the MPC based controllers with the lookahead horizons of $T = 1$ to $T = 8$, and the last one achieved by an optimal controller.*

FIGURE 2.4: *Optimal Service Replica Placement via Predictive Model Control from Hamoun Ghanbari, Przemyslaw Pawluk, Cornel Barna,* ***Marin Litoiu*** *at York University (Theme 2)*

2.5 An Architecture for Mitigation Low and Slow Application DDoS Attacks

Mark Shtern, Roni Sandel, Vasileios Theodorou, **Marin Litoiu** York U., and **Chris Bachalo** Juniper Networks

Distributed Denial of Service (DDoS) attacks are a growing threat to organizations. As defense mechanisms are advancing, hackers in turn are aiming at the application layer. For example, application layer Low and Slow Distributed Denial of Service attacks are becoming a serious issue because they are harder to detect due to low resource consumption.

In this poster, we propose a reference architecture that mitigates the Low and Slow DDoS attacks by utilizing Software Defined Infrastructure capabilities. Further, we propose two implementations of the reference architecture based on a Performance Model and Off-The-Shelf Component, respectively. We present the Shark Tank concept, a cluster under close scrutiny, where suspicious requests are redirected for further filtering.

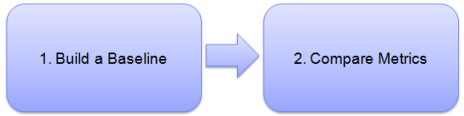
Towards Mitigation of Low and Slow DDoS Attacks

Mark Shtern, Roni Sandel, Marin Litoiu, Chris Bachalo, and Vasileios Theodorou
York University, Toronto, Canada

YORK UNIVERSITÉ UNIVERSITY

Performance Model-Based Approach

- **LSDDoS Detection Sensor**
  - Two phases:



- **Offline Approach**
- **Live Approach**
  - Advantage: Covers values that cannot be explored in an offline application
  - Disadvantage: Values may be collected when the system is under attack
- **Combination**

COTS Components

- **LSDDoS Detection Sensor:**
  - Junos DDoS Secure
    - Can monitor incoming traffic and outgoing traffic to and from the application
    - Uses heuristic algorithms for detection of attacks and suspicious users

JUNOS

- **Ganglia (Monitoring System)**
  - Scalable distributed monitoring system
  - Collects monitoring performance metrics
  - Gathers notifications of LSDDoS attacks

Ganglia

Both Approaches

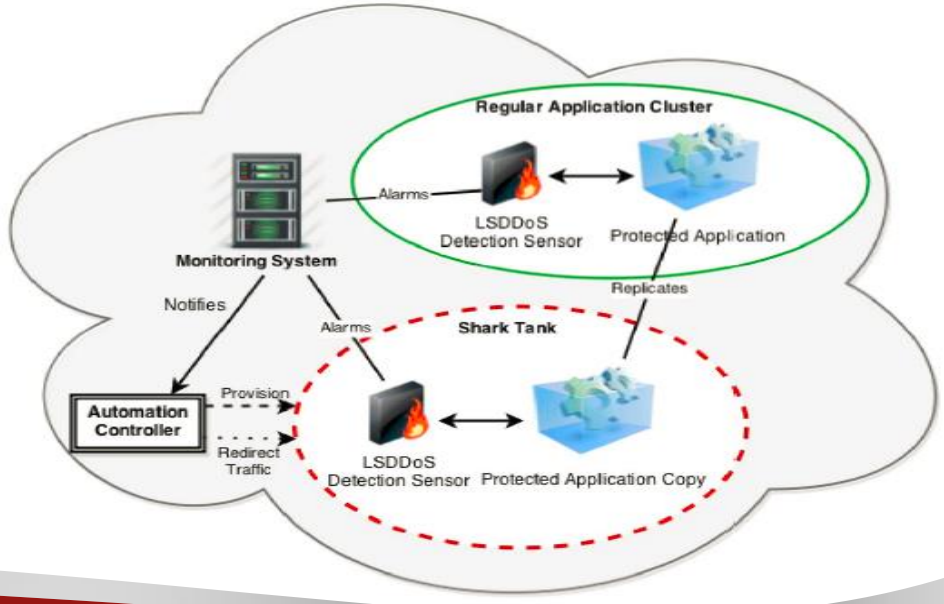
- **Automation Controller:**

Pattern Deployment System (PDS) :

- **XML-based Domain Specific Language (DSL)**
  - How the application will be deployed
- **Shark Tank will also be specified by DSL but with significantly reduced resources**
- **Monitoring system will be deployed with application**

Automation Manager:

- **Performance Model-Based Approach**
  - PDS' RESTful API
    - PDS notifies Open Flow controller to route all traffic from suspected IP address to Shark Tank's IP address
- **COTS Approach**
  - Juniper MX Router
    - PDS issues command to router to have suspect user traffic redirected to the Shark Tank



NSERC CRSNG Ontario SAVI

This work is funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP394424-10.

Web: http://www.ceraslabs.com
Email: mark@cse.yorku.ca, rsandel@yorku.ca, mlitoiu@yorku.ca, cbachalo@juniper.net, vasileios@essi.upc.edu

FIGURE 2.5: *An Architecture for Mark Shtern, Roni Sandel, Vasileios Theodorou, **Marin Litoiu** at York University and **Chris Bachalo** at Juniper Networks (Theme 2)*

2.6 Model-driven Elasticity and DoS Attack Mitigation in Cloud Environments

Cornel Barna, Mark Shtern, Hamoun Ghanbari, **Michael Smit**, **Marin Litoiu** York U.

Workloads for web applications can change rapidly. When the change is an increase in customers, a common adaptive approach to uphold SLAs is elasticity, the on-demand allocation of computing resources. However, application-level denial-of service (DoS) attacks can also cause changes in workload, and require an entirely different response. These two issues are often addressed separately (in both research and application).

This poster presents a model-driven adaptive management mechanism which can correctly scale a web application, mitigate a DoS attack, or both, based on an assessment of the business value of workload. This approach is enabled by modifying a layered queuing network model previously used to model data centers to also accurately predict short-term cloud behavior, despite cloud variability over time.

MODEL-DRIVEN ELASTICITY AND DOS ATTACK MITIGATION IN CLOUD ENVIRONMENTS

Cornel Barna, Mark Shtern, Michael Smit, Marin Litoiu, Hamoun Ghanbari

cornel@cse.yorku.ca, mark@cse.yorku.ca, msmit@dal.ca, mlitoiu@yorku.ca, hamoun@cse.yorku.ca

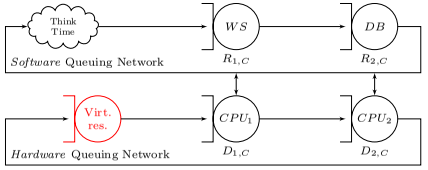
YORK UNIVERSITÉ UNIVERSITY

INTRODUCTION

The use of software-defined infrastructure enables the addition or removal of resources (computation, storage, networking, etc.) to or from a deployed web application at run-time in response to changes in workload or in the environment. Central to this elasticity is the use of mechanisms that autonomically decide when to make these changes.

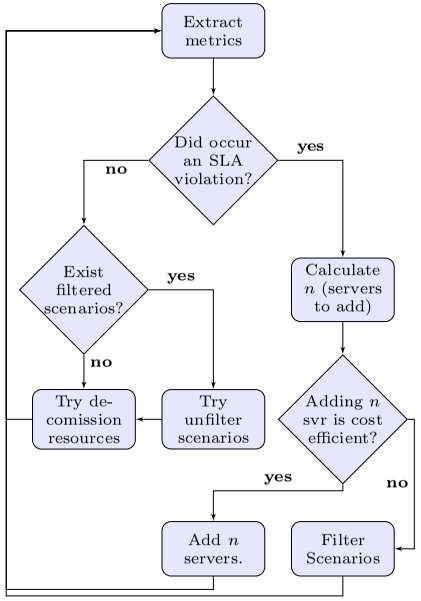
The problem becomes more complicated if the system is subject to a DDoS attack. During a DDoS attack, adding resources to handle the extra traffic leads to increase in cost without increase in income, while filtering all traffic means the non-malicious users don't receive service.

MODELING FOR CLOUDS

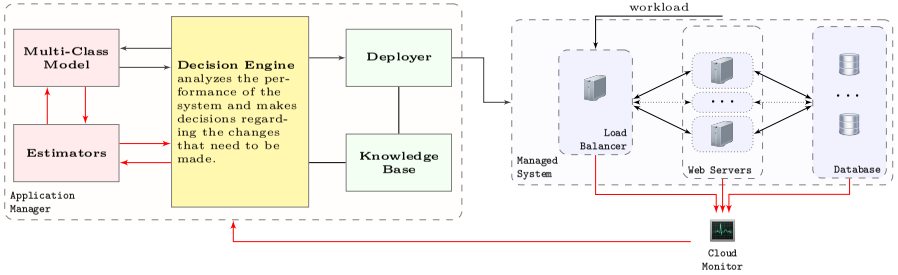


virtual resource—a resource designed to capture all undocumented work and delays. This resource is shared by all classes of traffic, it has no limit, and it is the first queue encountered by incoming requests.

METHOD



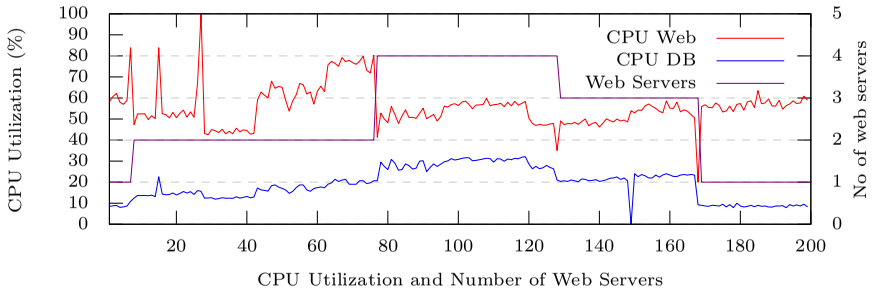
THE ARCHITECTURE



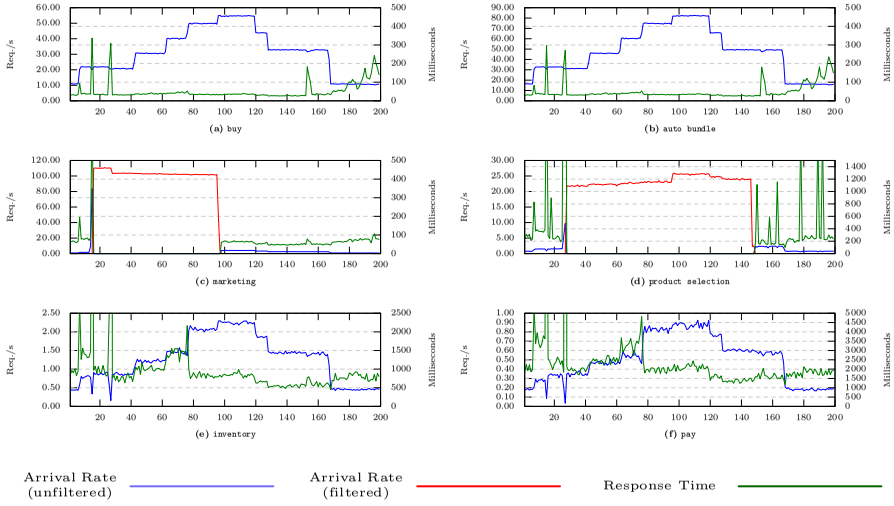
The system is continuously monitored; the performance metrics extracted are fed into the application manager. When the model becomes desynchronized, the application manager uses the `Estimators` component to estimate new values for the parameters.

RESULTS

The deployed web application (bookstore) had *6 classes of service*. We measured the CPU utilization on the database cluster (blue line) and on the web cluster (red line). The purple line shows the number of web servers (right Y-axis).



The experiment shows that the framework achieved elasticity in order to maintain SLOs for the desirable traffic, while detecting and redirecting undesirable traffic.



[1] BARNA, C., SHTERN M., SMIT M., GHANBARI H., LITOIU M. Model-driven Elasticity and DoS Attack Mitigation in Cloud Environments. In *11th International Conference on Autonomic Computing (ICAC)* (June 2014), pp. 13–24.

FIGURE 2.6: *Model-driven Elasticity and DoS Attack Mitigation in Cloud Environments from Cornel Barna, Mark Shtern, Hamoun Ghanbari,* ***Michael Smit, Marin Litoiu****, York University (Theme 2)*

2.7 Cloud bursting for MapReduce jobs: A dream or a reality?

Rizwan Mian, Mark Shtern, Saeed Zareian, **Marin Litoiu** York U.

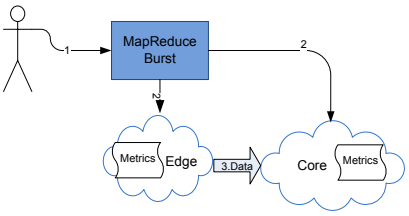
Cloud-bursting has been mostly explored for computational workloads, or assumes that the data already exists in the public clouds. We take a step towards data-intensive job bursting between clouds. In particular, we explore the practicality and usefulness of MapRedce (MR) job bursting between clouds assuming that there is high bandwidth between clouds. The experiments for MR bursting are conducted in a hierarchical cloud infrastructure, namely SAVI, in which edge and core are interconnected by high bandwidth links.

Towards Cloud Bursting for MapReduce Jobs

Rizwan Mian, Mark Shtern, Marin Litoiu, Saeed Zareian

MapReduce Cloud Bursting (MR Bursting)

- **Edge:** resource-constrained cloud
- **Core:** resource-abundant cloud
- Edge and core connected by high bandwidth link
- Burst into core **with Data**



Data Access Mechanisms

- **Local Access:** intra-cluster data access
- **Remote Access:** inter-cluster data access
- **Remote Copy:** inter-cluster data access and storage

Architecture

MR Burst Component

Controller
Cost Model
Performance Model

Bursting Heuristics

- **Greedy:** if edge idle submit job at edge, or else submit at core.
- **Cost-based:** determine costs of executing job at edge/core with different access mechanisms, and proceed with least costing option.

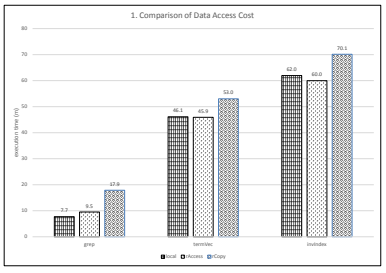
Rough Estimates

- **Local execution** = outstanding mappers + new mappers
- **Remote access execution** = new mappers + remote access
- **Remote copy execution** = new mappers + remote copy

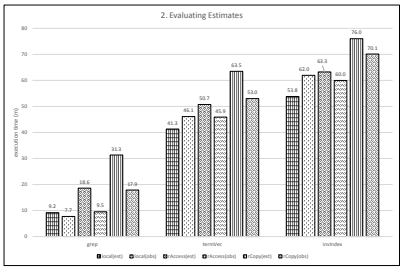
Experiments

- **Data set:** Free e-books 52GB (Gutenburg: www.gutenberg.ca)
- **Workloads:** distributed grep, term vector, inverted index (PUMA: sites.google.com/site/farazahmad/)
- Two separate Hadoop clusters of three VMs in core and edge.
- Workload profiling on 1GB of data to determine unit execution time of a single mapper (T)

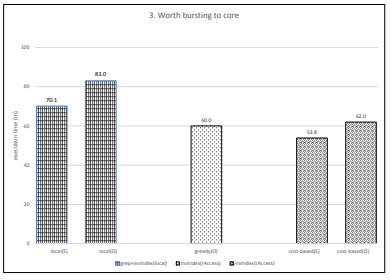
Benchmark	T (m)
grep	1.06
term vector	4.77
inverted index	6.21

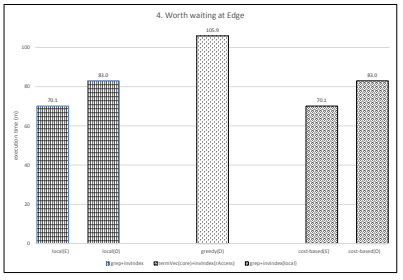


edge busy, core idle



edge busy, core busy





Discussion

- Observe benefits of MR bursting
- Estimated costs are similar to observed, but not always
- High bandwidth is a key to MR bursting

Future Work

- Extend cost models to include reducers, sorting and shuffle
- Explore performance models to predict cluster metrics
- Study MR bursting with different bandwidths
- Replicate data a priori to reduce cost of replication

FIGURE 2.7: *Cloud bursting for MapReduce jobs: A dream or a reality from Rizwan Mian, Mark Shtern, Saeed Zareian,* ***Marin Litoiu****, York University (Theme 2)*

2.8 Crowd-sourcing Sensor Data

Cornel Barna, Mark Shtern, Hamoun Ghanbari, **Michael Smit**, **Marin Litoiu** York U.

The Connected Vehicles and Smart Transportation (CVST) project is establishing a flexible and open application platform. It aims to integrate advanced wireless and sensor communications with mobile computing techniques in a cloud-based environment. As one of the practical steps, we have built a mobile application that interacts with a data management system to advise of travel conditions. Our aim is to alleviate transportation issues faced by commuters and government through analyzing the data obtained by crowd-sourcing of data from mobile users.

In our implementation, we use PlayFramework (http://www.playframework.com/) that uses Akka and Netty for fast and lightweight scalability and response, alongside HBase for scalability in the data layer. In addition, we use Apache Cordova in our mobile application for improved portability across different brands of mobile phones.

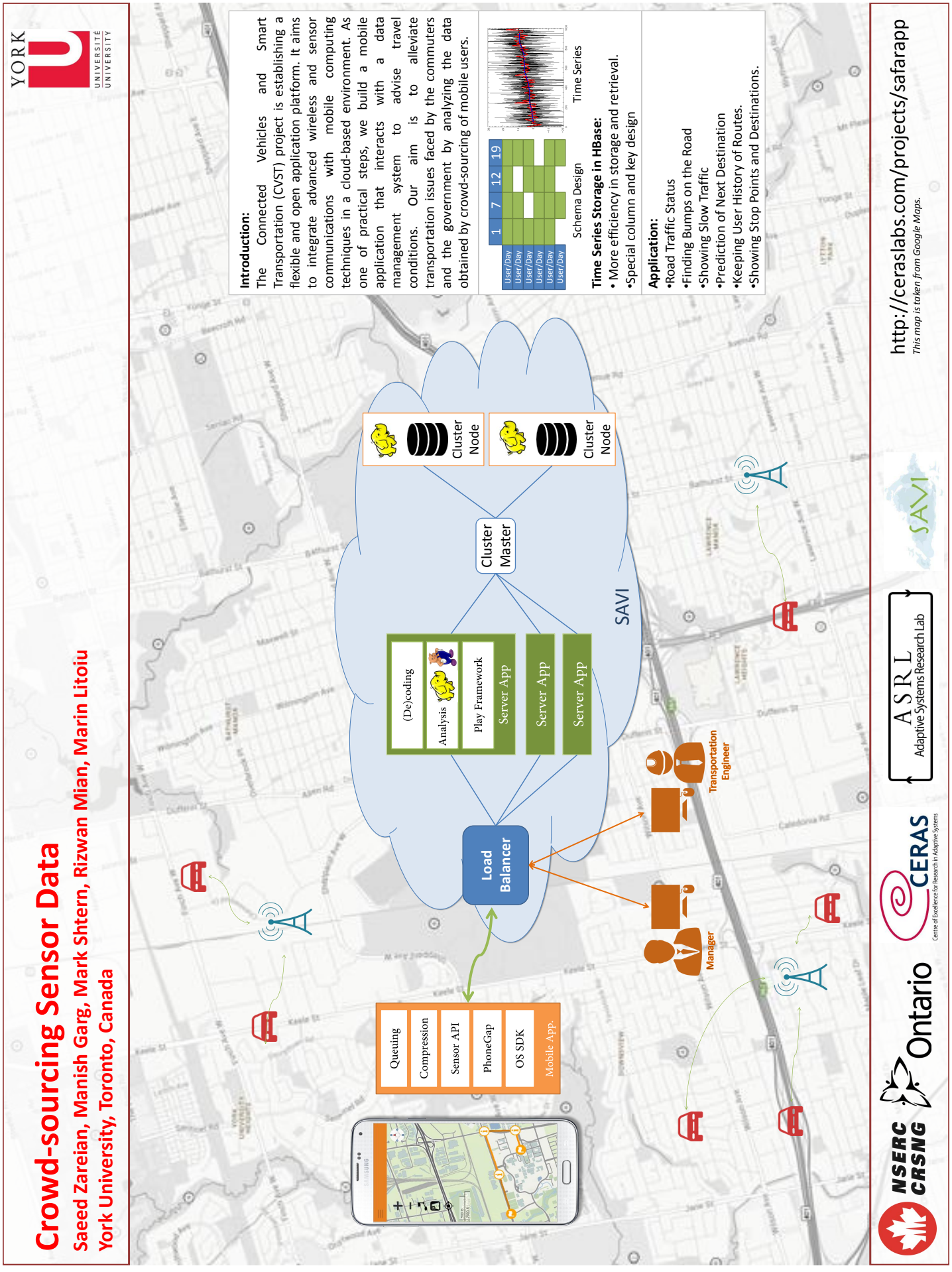


FIGURE 2.8: *Crowd-sourcing Sensor Data from Cornel Barna, Mark Shtern, Hamoun Ghanbari,* ***Michael Smit****,* ***Marin Litoiu*** *at York University (Theme 2)*

Page for Notes

Chapter 3

Theme 3 Smart Converged Edge

Research Team

Raouf Boutaba (Theme Lead), University of Waterloo
Yashar Ganjali, University of Toronto
Alberto Leon-Garcia, University of Toronto
Majid Ghaderi, University of Calgary
Carey Williamson, University of Calgary

3.1 On Satisfying Green SLAs in Distributed Clouds

Ahmed Amokrane, Mohamed Faten Zhani, Qi Zhang, Rami Langar, **Raouf Boutaba** U. Waterloo

With the massive adoption of cloud-based services, high energy consumption and carbon footprint of cloud infrastructures have become a major concern in the IT industry. Consequently, many governments and IT advisory organizations have urged IT stakeholders (i.e., cloud provider and cloud customers) to embrace green IT and regularly monitor and report their carbon emissions and at the same time, put in place efficient strategies and techniques to control the environmental impact of their infrastructures and/or applications.

Motivated by this growing trend, we are investigating how cloud providers can meet Service Level Agreements (SLAs) with green requirements. In such SLAs, a cloud customer requires from cloud providers that carbon emissions generated by the leased resources should not exceed a fixed threhold. We hence propose a resource management framework allowing cloud providers to provision resources in the form of Virtual Data Centers (VDCs) (i.e., a set of virtual machines and virtual links with guaranteed bandwidth) across a geo-distributed infrastructure with the aim of reducing operational costs and green SLA violation penalties. Extensive simulations show that the proposed solution maximizes the cloud provider's profit and minimizes the violation of green SLAs.

On Satisfying Green SLAs in Distributed Clouds

Ahmed Amokrane[1,2], Mohamed Faten Zhani[2], Qi Zhang[2], Rami Langar[1] and Raouf Boutaba[2]
[1]*Pierre and Marie Curie University, Paris VI, France,* [2]*University of Waterloo, ON*

Introduction

Context

- **Could Provider (CP)**
  - builds geographically distributed data centers
    - Variable electricity prices in different locations and over time
    - Variable availability of renewable power in different locations and over time
    - Variable carbon footprint for power from the grid depending on the location
  - Leases resources in the form of **Virtual Data Centers (VDCs)**
    - Virtual Machines (VMs) (CPU, memory and disk)
    - Virtual links (bandwidth and propagation delay)
- **Service Providers (SPs):** rent resources from the CP in the form of VDCs
- **Green Service Level Agreement (SLA)**:
  - Traditional SLA terms for VMs (CPU, memory) and virtual links (bandwidth, delay)
  - An upper limit of carbon emission for a reporting period (e.g., billing period)
  - SLA enforcement: penalty paid by the CP to the SP in case of violation of the carbon emission limit (e.g., partial refund of SP's bill)

Objectives from Cloud Provider's perspective

- Embed VDC requests across the distributed infrastructure
  - Maximize profit (revenue – cost and penalties)
  - Minimize carbon emissions
  - Dynamic reconfiguration through migration, follow the renewables
  - Minimize Green SLA violation penalty

Problem Formulation

System Model

- Decision variables: VM and virtual link mapping over time

$$x_{ik}^{j,t} = \begin{cases} 1 & \text{If the VM } k \text{ of the VDC } j \text{ is assigned} \\ & \text{assigned to data center } i \text{ during slot } t \\ 0 & \text{Otherwise.} \end{cases}$$

$$f_{e,e'}^{t} = \begin{cases} 1 & \text{If the link } e \in E \text{ is used to embed} \\ & \text{the virtual link } e' \in E^j \text{ during slot } t \\ 0 & \text{Otherwise.} \end{cases}$$

- Objective Function:

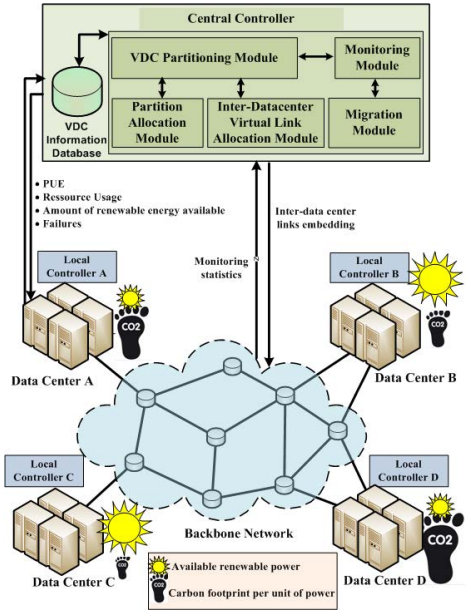
$$Maximize \quad \mathcal{R}_k - (\mathcal{D}_k + \mathcal{B}_k + \mathcal{M}_k + \mathcal{P}_k)$$

k: reporting period

R: Revenue, *D*: cost in data centers, *B*: inter-data center bandwidth cost, *M*: migration cost, *P*: penalty (SLA violation) cost

Proposed Framework

- **Central controller** manages the entire infrastructure.
  - VDC Partitioning Module: splits a VDC request into partitions
  - Partition Allocation Module: assigns partitions to data centers
  - Inter-data center Virtual Link Allocation Module: allocates virtual links in the backbone network
  - Migration Module: migrates partitions over time to optimize resource allocation
  - Monitoring Module: gathers various information from the infrastructure
- **Local controller:** manages internal resources inside a single data center



Resource allocation

- **Step 1 - VDC Partitioning:** Location-Aware *Louvain* Algorithm
  - Split the VDC requests into partitions with high intra-partition bandwidth demand and low inter-partition bandwidth demand
- **Step 2 - Partition Assignment :** Greedy assignment to data centers
- **Step 3 - Partition migration over time (every fixed reconfiguration period):**
  - Maximize the usage of renewables
  - Minimize carbon emissions
  - Minimize Green SLA violation penalty and electricity costs

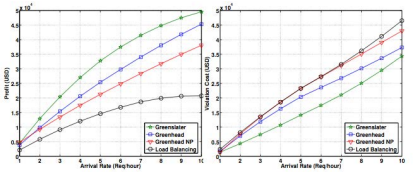
Simulation Results

Simulation Settings

- Simulated Infrastructure:
  - NSFNet backbone network with 4 data centers
  - Real traces of available renewable power and electricity prices
- VDC requests: Poisson arrivals, exponential lifetime, uniform number of VMs (10,50)
- Baseline: Greenhead, Greenhead without VDC partitioning, Load Balancing

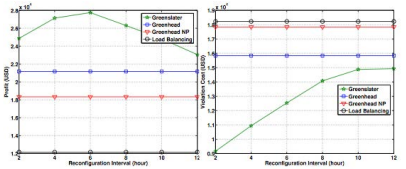
Simulation Results

- Impact of request arrival rates



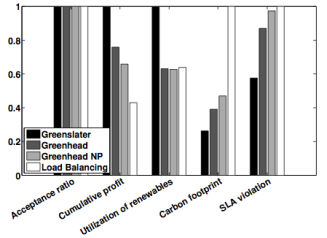
(a) Cumulative profit (b) SLA violation cost

- Impact of the reconfiguration interval



(a) Cumulative profit (b) SLA violation cost

- Other metrics



Conclusion

- The proposed VDC management framework is able to
  - Achieve high profit for the cloud provider
  - Maximize utilization of renewables
  - Minimize green SLA violation penalties

This work is funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP394424-10

FIGURE 3.1: *On Satisfying Green SLAs in Distributed Clouds from Ahmed Amokrane, Mohamed Faten Zhani, Qi Zhang, Rami Langar,* ***Raouf Boutaba*** *at University of Waterloo (Theme 3)*

3.2 CQNCR: Optimal VM Migration Planning in Cloud Data Centers

Md. Faizul Bari, Mohamed Faten Zhani, Qi Zhang, Reaz Ahmed, **Raouf Boutaba** U. Waterloo

With the proliferation of cloud computing, virtualization has become the cornerstone of modern data centers and an effective solution to reduce operational costs, maximize utilization and improve performance and reliability. One of the powerful features provided by virtualization is Virtual Machine (VM) migration, which facilitates moving workloads within the infrastructure to reach various performance objectives. As recent virtual resource management schemes are more reliant on this feature, a large number of VM migrations may be triggered simultaneously to optimize resource allocations. In this context, a challenging problem is to find an efficient migration plan, i.e., an optimal sequence in which migrations should be triggered in order to minimize the total migration time and impact on services.

In this work, we propose CQNCR (read as sequencer), an effective technique for determining the execution order of massive VM migrations within data centers. Specifically, given an initial and a target resource configuration, CQNCR sequences VM migrations that efficiently reach the final configuration with minimal time and impact on performance. Experiments show that CQNCR can significantly reduce total migration time by up to 35% and service downtime by up to 60%.

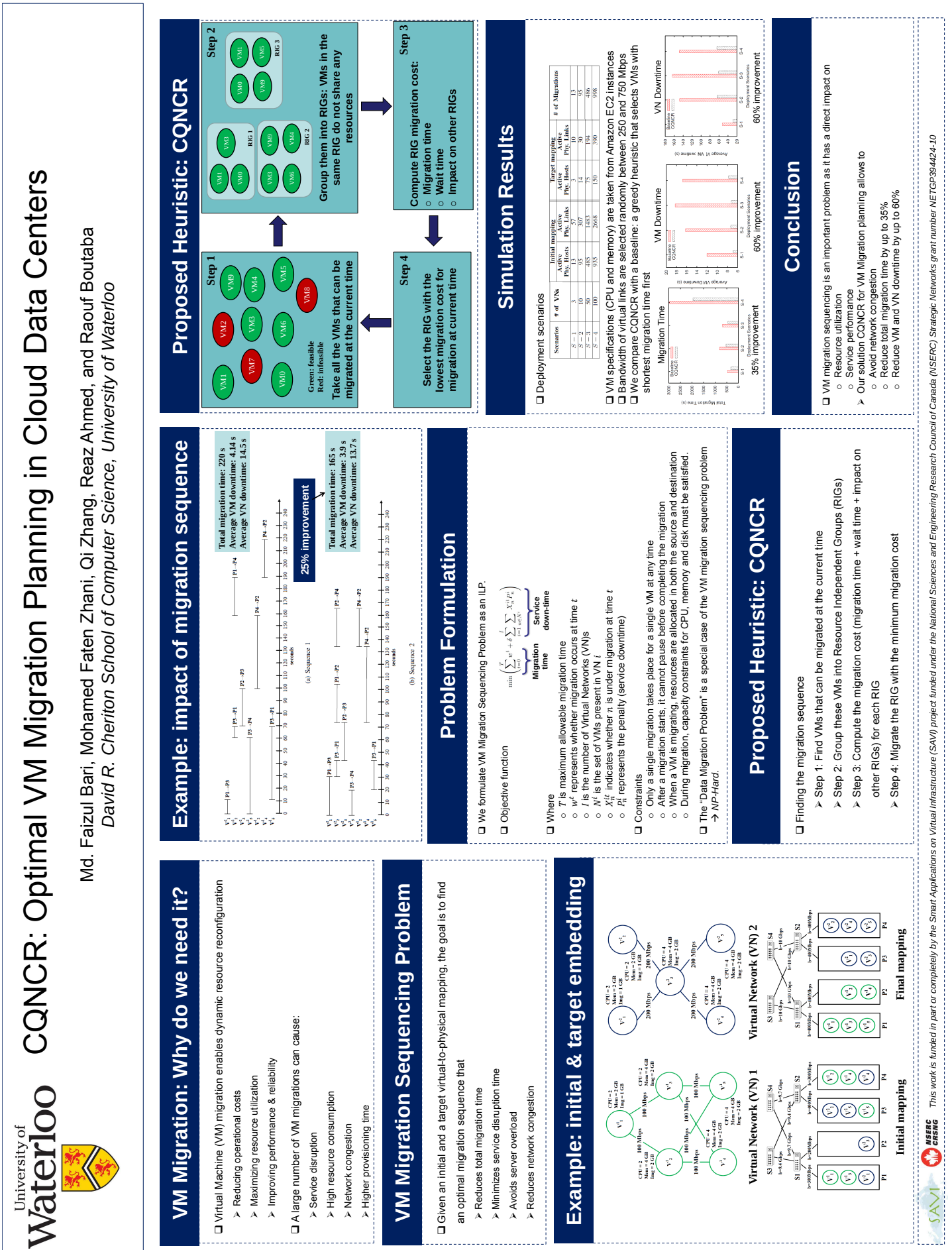


FIGURE 3.2: *CQNCR: Optimal VM Migration Planning in Cloud Data Centers from Md. Faizul Bari, Mohamed Faten Zhani, Qi Zhang, Reaz Ahmed,* ***Raouf Boutaba*** *at University of Waterloo (Theme 3)*

3.3 Design and Management of DOT: A Distributed OpenFlow Testbed

Arup Raton Roy, Md. Faizul Bari, Mohamed Faten Zhani, Reaz Ahmed, **Raouf Boutaba** U. Waterloo

With the growing adoption of Software Defined Networking (SDN), there is a compelling need for SDN emulators that facilitate experimentation with new SDN-based technologies. Unfortunately, Mininet (http://mininet.org/), the de facto standard emulator for software defined networks, is unable to scale with network size and traffic volume. The aim of this work is to fill this gap by presenting a low cost and scalable network emulator called Distributed OpenFlow Testbed (DOT). It can emulate large SDN deployments by distributing the workload over a cluster of compute nodes. Through extensive experiments, we show that DOT can overcome the limitations of Mininet and emulate larger networks. We also demonstrate the effectiveness of DOT on four Rocketfuel (http://rocketfuel.com/) topologies. DOT is available for public use and community-driven development at www.dothub.org .

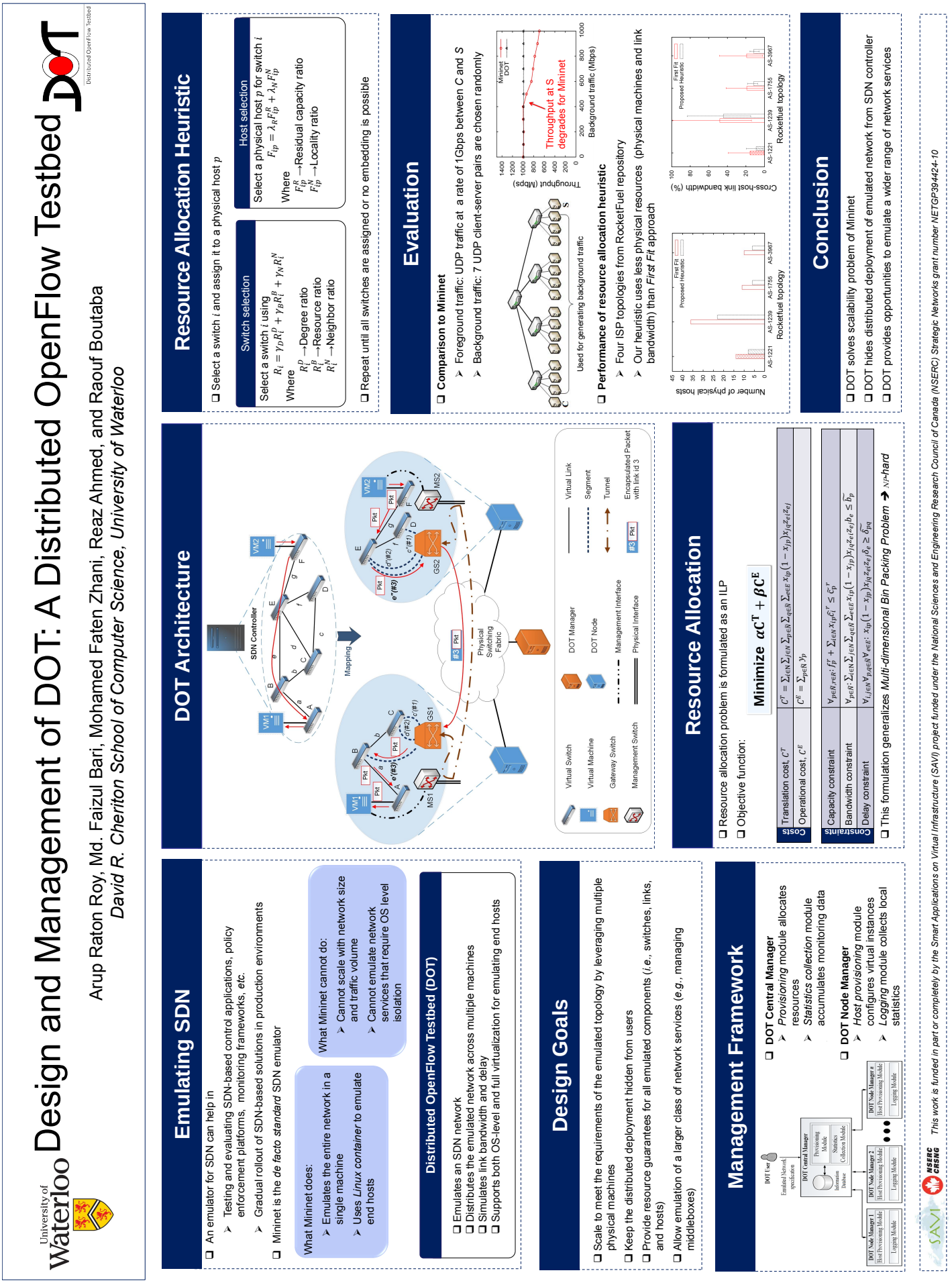


FIGURE 3.3: *Design and Management of DOT: A Distributed OpenFlow Testbed from Arup Raton Roy, Md. Faizul Bari, Mohamed Faten Zhani, Reaz Ahmed,* ***Raouf Boutaba*** *at University of Waterloo (Theme 3)*

3.4 A Simple Programming Model for Scalable SDN Control Applications

Submitted to IEICE Transactions on Electronics.
Soheil Hassas Yeganeh and **Yashar Ganjali** U. Toronto

Simplicity is a prominent advantage of Software-Defined Networking (SDN), and is often exemplified by implementing a complicated control logic as a simple control application deployed on a centralized controller. When it comes to practice, however, such simple control applications transform into complex logistics over distributed control platforms, since they need to tolerate eventual consistency (because existing control platforms favor availability) and implement complex coordination and partitioning mechanisms. As a result, distributed control applications are polluted with boilerplates of distributed programming that are usually more complicated than the control logic itself.

Here, we present a programming model that simplifies the development process of distributed applications. It is familiar and intuitive, yet generic enough to implement different communication patterns (such as Request/Response and Pub/Sub) and existing distributed controllers (such as ONIX and Kandoo). In summary, we have implemented a highly efficient control platform and our evaluations indicate that the proposed programming model does not impose an inherent scalability bottleneck.

A Simple Programming Model for Scalable SDN Control Applications

Soheil Hassas Yeganeh & Yashar Ganjali
University of Toronto

Introduction

EXISTING DISTRIBUTED CONTROL PLATFORMS

- Eventually consistent network graphs.
- Push all complexities to control applications.
- Difficult to program.

MOTIVATION

- Hiding these complexities behind a programming model.
- Programming Model: **map** + **recv**

OVERVIEW

- **map** maps a message to a set of IDs.
- Messages mapped to the **same ID** are received by the **same instance.**

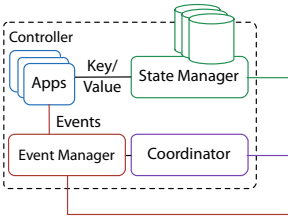
Design & Implementation

CONTROL APPLICATIONS

- Actors that exchange messages.
- Modeled as **map** + **recv**
- Implemented as centralized applications

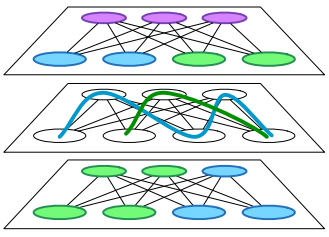
DISTRIBUTED CONTROL PLATFORM

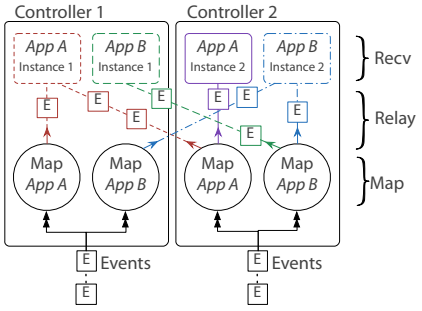
- Runs applications in a distributed fashion.
- Preserves correctness.
- Instruments applications at runtime.

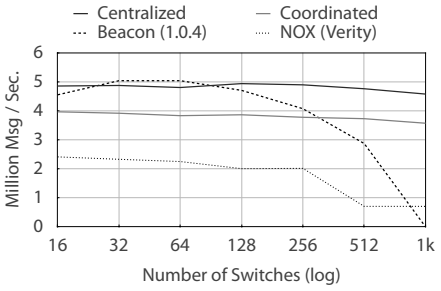


IMPLEMENTATION

- C++ (50k loc and + 30k loc generated code)







Use Cases

CENTRALIZED

maps messages to the same ID.

MASTER ELECTION

maps messages to switch ID.

KANDOO

maps messages to the controller ID.

ONIX

maps messages to NIB nodes.

PUB/SUB

maps messages to subscribers.

VIRTUAL NETWORK

maps messages to virtual network ID.

TRAFFIC ENGINEERING

Collectors are local & router is centralized.

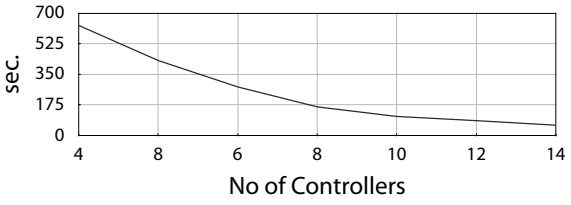
LOGICAL XBAR

maps messages to hierarchical IDs.

ROUTING

map route messages based on either prefixes, destinations or areas.

- Scales out for large networks.
- Can tolerate faults.
- Easy to adopt schemes such as Portland.



Simplicity / Scalability / Customizability / Automatic Placement / Fault Tolerance

FIGURE 3.4: *A Simple Programming Model for Scalable SDN Control Applications from Soheil Hassas Yeganeh and* ***Yashar Ganjali*** *at University of Waterloo (Theme 3)*

3.5 FleXam: Flexible Sampling Extension for Monitoring and Security Applications in OpenFlow

Sajad Shirali-Shahreza and **Yashar Ganjali** U. Toronto

Current OpenFlow specifications provide limited access to packet-level information such as packet content, making it very inefficient, if not impossible, to deploy security and monitoring applications as controller applications. In this poster, we present FleXam, a flexible sampling extension for OpenFlow designed to provide access to packet level information at the controller. The simplicity of FleXam makes it possible to implement it easily within OpenFlow switches and operate at line rate without requiring any additional memory. At the same time, its flexibility allows implementation of various monitoring and security applications in the controller, while maintaining balance between overhead and collected information details. FleXam realizes the advantages of both proactive and reactive routing schemes by providing a tunable trade-off between the visibility of individual flows, and the controller load. As an example, we show how FleXam can be used to implement a port scan detection application with an extremely low overhead.

FleXam: Flexible Sampling Extension for Monitoring and Security Applications in OpenFlow

Sajad Shirali-Shahreza, Yashar Ganjali
Department of Computer Science, University of Toronto
shirali@cs.toronto.edu, yganjali@cs.toronto.edu

OpenFlow Information Channels

Event-based Messages
- Sent by switches
- Usually deliver information about network structure and topology changes
- No information about active connections in the network

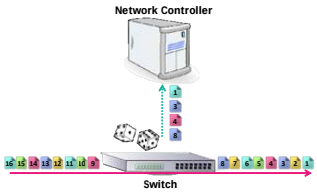
Flow Statistics
- Collected by switches
  - Received Packets, Received Bytes, Duration
- Pulled by the controller
- The only information channel about active flows in network
- No packet level information

Packet-in Messages
- Sent by switches
  - No matching rule for this packet
  - Send-to-controller action in matching rule
- Provide limited access to packet level information
- Switch may send only part of the packet to the controller

Security and Monitoring Application in OpenFlow

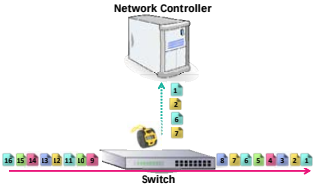
- Do not install any flow rule
  - Every packet will be sent to the controller
  - The controller tells switches what to do
- The controller sits on packet delivery path
  - Possible bottleneck
- Switch may buffer the packet
  - Only parts of each packet may reach the controller

- Send a copy of each packet to a monitoring device
  - Every packet will be sent entirely to the monitoring device
- High overhead for the network
- Monitoring device should handle large volumes of data
  - Even though it may only need packet headers



FleXam

- A flexible sampling extension for OpenFlow
  - A new action that can be assigned to any flow
- Enables the controller to define:
  - Which packets should be sampled
  - What parts of each packet should be selected
  - Where they should be sent to
- Simple enough to be done entirely in data path
- Flexible for different applications



Stochastic Sampling
- Select each packet of the flow with a probability of ρ
- Implementation:
  - Generate a random number
  - Sample the packet if it is less than ρ

Deterministic Sampling
- Select m consecutive packets out of every k consecutive packets, skipping the first δ
  - $m=1$ is the normal one out of k, or every k^{th} packet sampling
- m>1, large k $\rightarrow$ only sample first m packets
  - Suitable for applications such as traffic classification
- δ >0 $\rightarrow$ exclude small flows (mice flows)
- Implementation:
  - Use received packet counter and sample packet if: ((Received_Packet_Counter-δ) % k)<m,

Advantages

- Easy implementation in the switch; no major changes in hardware or software
- No overhead for flows without sampling
- Small networks: samples can be processed in the controller
  - Simple security applications
- Large networks: distribute sample processing
- Network overhead tunable by the controller

Sample Application: Port Scan Detection

- Randomly sample packets from different flows
  - TCPSYN packet is sampled
    - Install a deterministic sampling rule to sample the response ACK packet
    - If not received the ACK response, mark this as a failed connection
  - Other packets are sampled: a successful connection
- Install rules to exclude heavy flows from sampling
- Details in our HotI 2013 paper "Efficient Implementation of Security Applications in OpenFlow Controller with FleXam"

FIGURE 3.5: *FleXam: Flexible Sampling Extension for Monitoring and Security Applications in OpenFlow from Sajad Shirali-Shahreza and* ***Yashar Ganjali*** *at University of Toronto (Theme 3)*

3.6 Dynamic Virtual Infrastructure Provisioning in Geographically Distributed Clouds

Qi Zhang and **Alberto Leon-Garcia** U. Toronto

Large-scale online services today often span multiple geographically distributed domains and require significant storage, compute and networking resources. Furthermore, these services often need to be scaled up and down dynamically according to service demand fluctuation. To facilitate efficient resource allocation while allowing multiple online services to share to physical hosting infrastructure (e.g. data centers), there is a emerging trend towards allocating resources to service applications in the form of virtual infrastructures (VI) that consist of virtual machines, virtual routers and switches interconnected by virtual links. However, despite extensive study of virtual infrastructure scheduling algorithms, existing work has not studied the problem of provisioning VIs dynamically according to demand fluctuations. To address limitation, in this work we present a framework for virtual infrastructure provisioning that adjusts virtual infrastructure resource allocation according to demand fluctuations, while satisfying performance requirements. We demonstrate the effectiveness of our framework through simulations in realistic application scenarios.

Dynamic Service Provisioning in Geographically Distributed Clouds

Qi Zhang and Alberto Leon-Garcia

Department of Electrical and Computer Engineering, University of Toronto

Contact Information:
Electrical and Computer Engineering
University of Toronto
Phone: +1 (647) 376 6935
Email: ql.zhang@utoronto.ca

1 Introduction

- Service providers today deploy services across geographically distributed data centers
- To achieve performance, management and security goals, it is necessary to allocate virtual networks in addition to server resources in the form of **virtual infrastructures (VIs)**.
  - Guaranteed network resources in addition to server resources
- However, as service demand may fluctuate over-time, there is a need to scale VIs according to demand fluctuation
- **Dynamic VI scaling problem**: Given a virtual topology graph that consists of servers and the interconnect virtual network
  - Decide how many servers to be allocated in each data center
  - Decide how to embed the interconnect virtual network
  - Adapting VI embedding according to demand fluctuation and migration cost
- Related Work
  - **Virtual Network Embedding**: focus on static topologies only
  - **Service Placement**: Does not consider bandwidth requirement
  - **Autonomic Server Provisioning**: Focus on single data center scenarios only
- Research challenges:
  - How to define the VI request that enables dynamic scaling?
  - How to perform dynamic VI provisioning at run-time?

2 Request Specification

- Each *request* consists of multiple services I
- Each service is described by a *service graph* $\mathscr{G}^i = (N^i, E^i)$
- Demand originates from $\bar{A} \subseteq \bar{N}$ in the physical topology
- Server specifications
  - Each server $n \in N^i$ has resource requirement c^{nr} for each resource $r \in R$
  - Each server $n \in N^i$ can be placed in locations $\bar{N}^n \subseteq \bar{N}$
  - Each link $l \in E^i$ has a bandwidth requirement c^l for processing a single request of type i.
- Delay requirements
  - There is a set of paths $\mathscr{P}^l$ that have delay requirements
  - $p \in \mathscr{P}^l$ has a maximum delay threshold d^p.
  - Each link has a network delay d^l
  - Each server $n \in N^i$ has service capacity C^n and average delay d^n
- Inter-server communication requirement
  - Instances of the same server n each has bandwidth requirement $\beta = af + b$ to its peers, where and f is the total demand processed at n.

3 An Example

- We consider a 3-tier web application with an authentication service

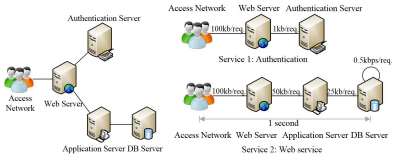


Figure 1: Example App.

Figure 2: Service Request

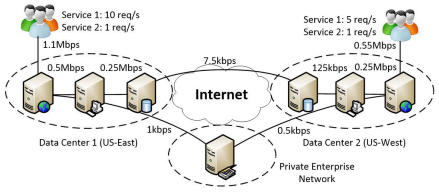


Figure 3: Resulting Embedding

4 Problem Formulation

- The goal of the dynamic VI scaling problem is to minimize

$$\min \quad C+R$$

where

$$C = \sum_{i\in I}\sum_{g\in G^i}\sum_{l\in L^i}\sum_{\bar{n}\in\bar{N}}\left(\sum_{\tau\in T} x_l^{\tau\bar{n}}\gamma^{\tau\bar{n}} + \left(\sum_{\bar{m}\in\bar{M}} f^{gl\bar{m}\bar{n}}\gamma^l + \sum_{g'\in G^i\setminus\{g\}}\sum_{n\in N^i} f^{gg'n\bar{m}\bar{n}}\right)\right)$$

$$R = \sum_{i\in I}\sum_{g\in G^i}\sum_{l\in L^i}\sum_{\bar{n}\in\bar{N}}\left(\sum_{\tau\in T} u^{\tau\bar{n}}c^{n\tau} + \sum_{\bar{m}\in\bar{M}}\delta^{gl\bar{m}\bar{n}}c^{\bar{m}\bar{n}}\right)$$

- Flow constraints

$$\sum_{(\bar{m},\bar{n})\in\bar{L}} f^{gl\bar{m}\bar{n}} - \sum_{(\bar{n},\bar{m})\in\bar{L}} f_n^{gl\bar{m}\bar{n}} = 0 \quad \forall i\in I, l\in E^i, g\in G^i, \bar{n}\in\bar{N}\setminus\{h^{gs^l}, h^{gd^l}\}$$

$$\frac{1}{w^{gl}}\sum_{(\bar{m},\bar{n})\in\bar{L}} f^{gl\bar{m}\bar{n}} = f^{g} \quad \forall i\in I, l\in E^i, g\in G^i, h^{gd^l}=\bar{n} \tag{1}$$

$$\frac{1}{w^{gl}}\sum_{(\bar{m},\bar{n})\in\bar{L}} f^{gl\bar{m}\bar{n}} = f^{g} \quad \forall i\in I, l\in E^i, g\in G^i, h^{gd^l}=\bar{n} \tag{2}$$

$$\sum_{g\in G^i:a^g=\bar{a}} f^g = D^{i\bar{a}} \quad i\in I, \bar{a}\in\bar{A}^i \tag{3}$$

- Capacity constraints

$$\sum_{i\in I}\sum_{n\in N^i} x^{\tau\bar{n}}c_n^{r\tau} \le \rho_{\bar{n}}^r \quad \forall\bar{n}\in\bar{N}, r\in R \tag{4}$$

$$\sum_{i\in I}\sum_{g\in G^i}\left(f^{gsl'd'} + \sum_{g'\in G^i\setminus\{g\}}\sum_{m\in N^i} f^{gg'md'}\right) \le b_{\bar{l}} \quad \forall\bar{l}\in\bar{L} \tag{5}$$

- Delay constraints

$$\frac{f^{gl\bar{m}\bar{n}}}{\varepsilon} \ge y^{gl\bar{m}\bar{n}} \ge \frac{f^{gl\bar{m}\bar{n}}}{c^l} \quad \forall g,l,(\bar{m},\bar{n})\in\bar{L} \tag{6}$$

$$\pi^{gl,\bar{n}} = 0 \quad \forall i\in I, g\in G^i, g(p_1)=\bar{n} \tag{7}$$

$$\pi^{gl\bar{n}} \ge \pi^{gl\bar{m}} + y^{gl\bar{m}\bar{n}}d^{gl\bar{m}\bar{n}} \quad \forall i\in I, g\in G^i, (\bar{m},\bar{n})\in\bar{L} \tag{8}$$

$$\pi^{gl\bar{n}} \ge \pi^{gl\bar{n}} + y^{gl\bar{m}\bar{n}}q^{\bar{n}}\Big(\sum_{(\bar{m},\bar{n})\in\bar{L}} f^{gl\bar{m}\bar{n}}\Big) \quad \forall i\in I, g\in G^i, (\bar{m},\bar{n})\in\bar{L}, h^{gd^l}=\bar{n} \tag{9}$$

$$\pi^{gl\bar{n}} \le d^p \quad \forall i\in I, l\in G^i, p\in\mathscr{P}^i, l\in G^i, \bar{n}\in\bar{N} \tag{10}$$

- This problem is clearly $\mathscr{NP}$-hard as it generalizes the virtual network embedding problem

5 Solution Algorithm

- The key challenge is to find a trade-off between satisfying delay requirement and minimizing solution cost
  - Placing servers based only on cost may not lead to a feasible solution
  - Placing servers based on delay may lead to a high cost solution
- Our approach is to try different values for the trade-off factor w
  - For each trade-off factor w we can solve the assignment problem using greedy or multi-commodity flow algorithms
  - Find w that produces the best solution

```
1:  w ← 0
2:  for i ∈ I do
3:    while ∃P ∈ P^i that is not embedded do
4:      F^{l_0 n̄} ← D_{n̄}
5:      for j = 1 to |P| do
6:        Ū ← N̄^{p_j}
7:        while ∃F^{l_{j-1} n̄} > 0 and Ū ≠ ∅ do
8:          Find n̄ and {f^{l_{j-1} n̄ n̄}} with lowest C_{p_j n̄}
9:          F^{l_{j-1} n̄} ← F^{l_{j-1} n̄} − {f^{l_{j-1} n̄ n̄}}
10:         F^{l_j n̄} ← F^{l_j n̄} + {f^{l_{j-1} n̄ n̄}}
11:       if ∃F^{l_{j-1} n̄} > 0 and Ū = ∅ then
12:         return VI cannot be embedded
13: for i ∈ I do
14:   for n ∈ N^i that is not embedded and ∃l ∈ E^i : d(l) == n, s(l) is
      embedded do
15:     for l ∈ E^i : d(l) == n, s(l) is embedded do
16:       m ← s(l)
17:       F^{l n̄} ← (w^{gl} / Σ_{n'∈N̄} f^{l n' n̄}) f^{l n' n̄}
18:       Ū ← N̄^{p_j}
19:       while ∃F^{l n̄} > 0 and Ū ≠ ∅ do
20:         Find n̄ and {f^{l n̄ n̄}} with lowest C_n n̄
21:         F^{l n̄} ← F^{l n̄} − {f^{l n̄ n̄}}
22:       if ∃F^{l n̄} > 0 and Ū = ∅ then
23:         return VI cannot be embedded
```

6 Experiment Results

- We implemented our algorithm and compared it with two heuristics.
  - **min-cost heuristic** places each node based solely on resource cost
  - **min-distance heuristic** place each node to minimize total distance.
- Topology are randomly generated with demand fluctuates over time
- The profit of a VI is determined by the service revenue (total demand * revenue per request) minus embedding cost.

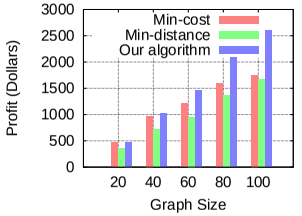


Figure 4: *Profit Earned*

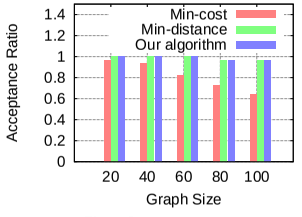


Figure 5: *Acceptance Rate*

7 Conclusion

- Allocating resources in the form of VIs has gained popularity
- As service demand may fluctuate over time across multiple geographical regions, there is a need to
  - Servers may be replicated across multiple data centers with different delay and resource costs.
  - Both server allocation and network allocation may change over time to accommodate demand fluctuations.
- We provide an algorithm to solve dynamic VI scaling problem.
- Experiments show our algorithm achieves 27% gain compared to naïve solutions.

FIGURE 3.6: *Dynamic Virtual Infrastructure Provisioning in Geographically Distributed Clouds from Qi Zhang and **Alberto Leon-Garcia** at University of Toronto (Theme 3)*

3.7 Online Algorithms for Energy Cost Minimization in Cellular Networks

Ali Abbasi and **Majid Ghaderi** U. Calgary

Dynamic base station activation and transmission power control are key mechanisms to reduce energy consumption in cellular networks. In this work, we consider employing these methods for the purpose of minimizing long-term energy cost in cellular networks. Based on the two-timescale Lyapunov optimization technique, we formulate an online control problem to ensure that we can achieve minimal energy cost while stabilizing use queues. While the control problem can be solved in a centralized manner, we limit our attention to distributed solutions which are highly attractive in the design of next generation mobile networks. Due to the combinatorial nature of the problem and the complex relation of achievable rates to interfering signals, the problem is non-convex. Consequently, conventional duality methods cannot be employed to achieve the distributed solution.

Thus, we propose and design a distributed solution for the problem based on Gibbs sampling method. The proposed algorithm can be implemented in a fully distributed manner, does not depend on the convexity or continuity of the energy cost functions, and guarantees solution optimality. Numerical results are provided to demonstrate the behavior of the solution in some example network scenarios.

Online Algorithms for Energy Cost Minimization in Cellular Networks

Ali Abbasi and Majid Ghaderi, University of Calgary

NSERC CRSNG

SAVI

Motivation

Cellular data traffic has massively increased in recent years.
To cope with the traffic demand, cellular base stations are deployed more densely which results in

- high energy consumption.
- high coverage overlap.

While cellular traffic exhibits periodic behavior, the energy consumption approximately remains the same.

Base stations are deployed to satisfy the peak traffic demand, while keeping them active all the time.

Current BSs consume about 50% of their peak energy in idle mode.
Power control does not lead to high energy saving.
Solution: Dynamic BS activation

Problem

Goal: minimizing the *long-term energy cost* of operating a cellular network using dynamic BS activation.

Requirements:

- The solution should only rely on the information that is readily available to the network, *e.g.*, user queue backlogs and average power price.
- Due to importance of self-organization and self-optimization in future cellular networks, we focus on distributed solutions.

The problem is modeled following the framework of Lyapunov optimization.

Gibbs sampling is employed to devise a distributed solution that can handle the nonconvexity of the problem.

Solution Algorithm

Queue $Q_j(t)$ is kept for each user u_j at its associated base station.

Queues evolve according to the following queuing dynamic,

$$Q_j(t+1) = [Q_j(t) - \sum_{b_i \in \mathcal{B}} y_i(t) a_{ij}(t) r_j(t)]^+ + w_j(t),$$

where $w_j(t)$ is the arrival workload for user u_j and $y_i(t)$ is a binary variable that indicates activation of b_i at t.

The objective is to minimize

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{Cost(\tau)\},$$

subject to network stability condition

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{j=1}^{m} \mathbb{E}\{Q_j(\tau)\} < \infty\,.$$

To solve the problem, the Lyapunov function $L(t)$ is defined as $L(t) = \sum_{j=1}^{m} \frac{1}{2}[Q_j(t)]^2$.

The goal is to push the system towards a lower backlog state and minimize the energy cost over each time frame. This goal is captured by the following drift-plus-penalty expression

$$\mathbb{E}\{L(t+T) - L(t) | \boldsymbol{Q}(t) + V \cdot \mathbb{E}\{ \sum_{\tau=t}^{t+T-1} Cost(\tau)\},$$

where the parameter V is chosen so as to control the trade-off between energy cost and congestion.

To minimize the drift-plus-penalty, it is proved that net utility of the system which is defined as follows

$$\mathbb{E}\{ \sum_{\tau=t}^{t+T-1} \sum_{b_i \in \mathcal{B}} y_i(t) [\sum_{u_j \in \mathcal{U}_i} Q_j(t) a_{ij}(t) r_j(t) - V C_i(\tau)] | \boldsymbol{Q}(t)\}$$

should be maximized. To do so, at the end of each frame, every BS b_i updates its power vector $\boldsymbol{P}_i$ to a new $\boldsymbol{P}_i'$ based on the Gibbs probability as follows

$$\mathbb{P}\{\boldsymbol{P}_i \to \boldsymbol{P}_i'\} = \frac{\exp(N_q([\boldsymbol{P}_i', \boldsymbol{P}\backslash\boldsymbol{P}_i])/\Theta)}{\sum_{\boldsymbol{P}_i'' \in \mathcal{P}_i} \exp(N_q([\boldsymbol{P}_i'', \boldsymbol{P}\backslash\boldsymbol{P}_i])/\Theta)},$$

where q is the clique in the neighborhood graph of the network that contains b_i, $\boldsymbol{P}$ is the vector of all power vectors, Θ is a constant called the temperature, and N_q is defined as follows

$$N_q(f) = \sum_{b_i \in q} \mathbb{E}\Big\{ \sum_{\tau=t}^{t+T-1} y_i(t) \times [\sum_{u_j \in \mathcal{U}_i} Q_j(t) a_{ij}(t) r_j(t) - V C_i(\tau)] | \boldsymbol{Q}(t) \Big\}$$

Any change to the local state of a BS is carried out in two steps:

1. First, the power vector is updated. This may also include activating or shutting down a BS.
2. Second, the time fraction vector that results in the optimal net utility for the new power vector is chosen as the new time fraction vector.

Model

We consider downlink of an OFDMA-based cellular network, *e.g.*, downlink in an LTE network. The system consists of
the set $\mathcal{B} = \{b_1, \ldots, b_n\}$ of BSs,
the set $\mathcal{U} = \{u_1, \ldots, u_m\}$ of users,
and the $\mathcal{H} = \{h_1, \ldots, h_k\}$ of subcarriers.

BS activation is performed at a timescale that is different from other network operations, *e.g.*, user scheduling. Time is divided into frames of size T timeslots. The activation/de-activation decisions are made at the beginning of each time frame.

SINR of user u_j when served by BS b_i on subcarrier h_k is given by

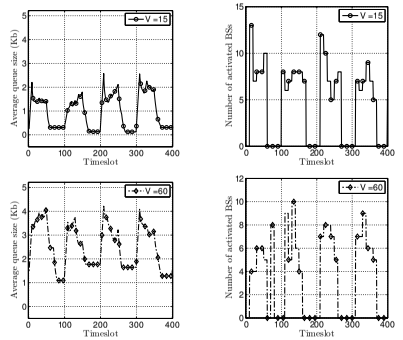
$$\text{SINR}_{ijk}(t) = \frac{P_{ik}(t) \cdot g_{ijk}}{\sum_{b_{i'} \neq b_i} P_{i'k}(t) \cdot g_{i'jk} + \eta},$$

where η is the noise power, $P_{ik}(t)$ is the power allocated to k-th subcarrier from b_i at timeslot t, and g_{ijk} is the power gain between b_i and u_j on h_k. The achievable rate of user u_j on h_k is obtained as

$$R(\text{SINR}_{ijk}(t)) = \log(1 + \text{SINR}_{ijk}(t))\,.$$

Numerical Results

A network of 25 BSs is considered. The behavior of GSA is studied in terms of energy cost and delay trade-off. These parameters can be approximated via the number of active BSs and average queue sizes respectively. Each frame consists of 10 timeslots. Frames are divided into framesets of size 5. During even-numbered framesets (timeslots 0-49,100-149,...) user queues receive data while there is no data arrival in odd-numbered framesets. By increasing V, the average queue sizes increases and the average number of active BSs decreases.



Model (Cont'd)

Then, the total received rate of u_j is given by

$$r_j(t) = \sum_{b_i \in \mathcal{B}_j} \sum_{h_k \in \mathcal{H}} a_{ij}(t) \tau_{ijk}(t) R(\text{SINR}_{ijk}(t)),$$

where $\tau_{ijk}(t)$ denotes the fraction of time that subcarrier h_k is allocated to user u_j and a_{ij} indicates association of u_j to b_i.

Users associate to the BSs that provide them with the highest *Reference Signal Received Power*.

The transmission power of a BS can vary from 0 to P_{max} in small discrete steps of size δ, *i.e.*, $0, \delta, 2\delta, \ldots, P_{max}$.

If BS b_i consumes the total power $P_{b_i}(t)$, the energy *cost* incurred by b_i is given by

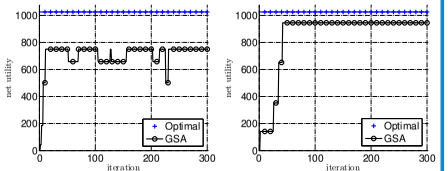
$$C_i(t) = C_P(t) \cdot P_{b_i}(t), \qquad (1)$$

where $C_p(t)$ is the energy price at t. The energy cost of the system is the sum of the energy costs of all base stations, *i.e.*,

$$Cost(t) = \sum_{b_i \in \mathcal{B}} C_i(t)\,.$$

Numerical Results

The performance of Gibbs sampling algorithm (GSA) is compared against the optimal solution in terms of the achieved net utility. The figure demonstrates the GSA results when temperature Θ is set to 50 and 0.1 respectively. As seen in the figure, the GSA becomes very close to the optimal when a small value is used for Θ.



Θ = 50. Θ = 0.01.

Achnowledgement

This work is funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP394424-10.

Figure 3.7: *Online Algorithms for Energy Cost Minimization in Cellular Networks from Ali Abbasi and* ***Majid Ghaderi*** *at University of Calgary (Theme 3)*

3.8 Greenhead: Virtual Data Center Embedding Across Distributed Infrastructures

Aimal Khan, Ahmed Amokrane, Arup Raton Roy, Mohamed Faten Zhani, Maissa Jabri, Qi Zhang, Rami Langar, **Raouf Boutaba** U. Waterloo

In our recent work, we proposed Greenhead, a holistic resource management framework for embedding Virtual Data Centers (VDCs) across distributed infrastructure. Greenhead is able to achieve multiple goals including revenue maximization, operational costs reduction, energy efficiency, and green IT, or to simply satisfy geographic location constraints of the VDCs. In this demo, we show how SAVI users can define their VDC specifications and how Greenhead allocates the required resources across the smart edges. In particular, we implemented Greenhead as a testbed-wide management module that communicates with local smart edge managers (VDCPlanner) to efficiently allocate computing and networking resources while satisfying geographic location constraints (i.e., proximity of some VMs to end users).

Greenhead: Virtual Data Center Embedding Across Distributed Infrastructures

Aimal Khan†, Ahmed Amokrane*, Arup Raton Roy†, Maissa Jabri†, Mohamed Faten Zhani†,
Qi Zhang†, Rami Langar* and Raouf Boutaba†
†*University of Waterloo* **Pierre and Marie Curie University, Paris VI*

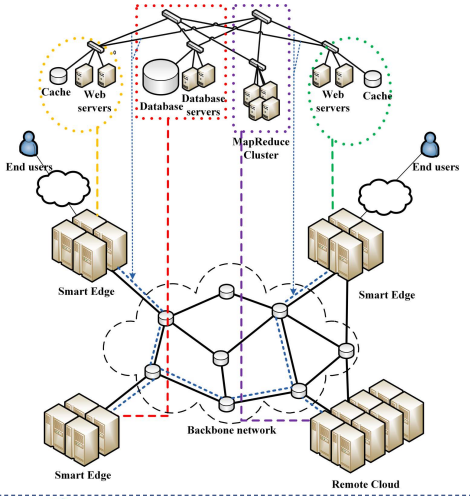
Introduction

Objectives

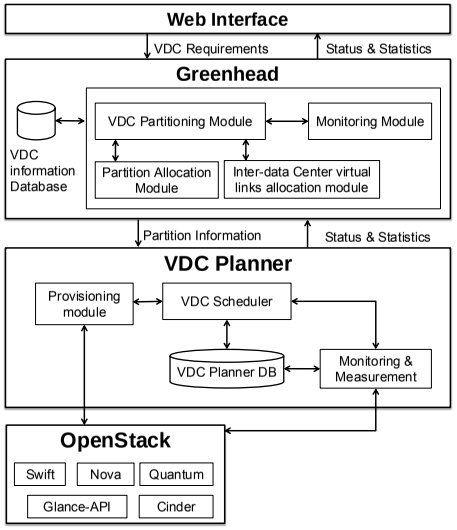
- Allow Infrastructure Providers (**InPs**) to allocate Virtual Data Centers (**VDCs**) over a geographically distributed infrastructure (e.g., SAVI testbed).

Overview

- **Greenhead** enables infrastructure providers to provision VDCs over a geographically distributed infrastructure and provides efficient embedding based on various requirements (e.g., latency) and goals (e.g., energy efficiency, backbone utilization).
- **Virtual Data Center (VDC) Planner** manages computing and networking resource allocation within a single smart edge.



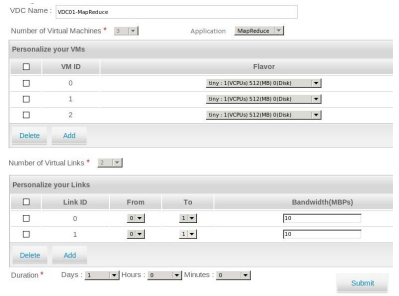
Block Diagram



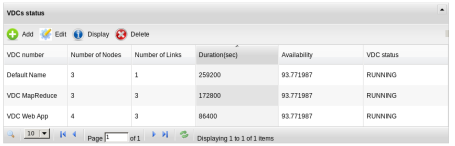
- **Greenhead**
  - **VDC Partitioning Module** splits the VDC request into partitions and assigns each partition to a VDC Planner instance.
  - **Monitoring Module** tracks the current status of the distributed infrastructure.
- **VDC Planner**
  - **VDC Scheduler** allocates resources required for each partition as requested by Greenhead. It invokes OpenStack APIs via the **Provisioning Module** to allocate VMs and Links.
  - **Monitoring & Measurement Module** keeps track of the status of the physical and virtual resources in a smart edge.

User Interface

- Creation of a new VDC:



- VDC Management:



Future Work

- Add fault-tolerance and failure management functionality.
- Integrate efficient VM migration management tools.

References

- M. F. Zhani, Q. Zhang, G. Simon, and R. Boutaba, VDC Planner: Dynamic Migration-Aware Virtual Data Center Embedding for Clouds. In IFIP/IEEE Integrated Network Management Symposium (IM 2013), Ghent, Belgium, May 2013.
- A. Amokrane, M. F. Zhani, R. Langar, R. Boutaba, and G. Pujolle. Greenhead: Virtual data center embedding across distributed infrastructures. IEEE Transactions on Cloud Computing, vol. 1, no. 1, January-June 2013.

This work is funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP394424-10

FIGURE 3.8: *Greenhead: Virtual Data Center Embedding Across Distributed Infrastructures from Aimal Khan, Ahmed Amokrane, Arup Raton Roy, Mohamed Faten Zhani, Maissa Jabri, Qi Zhang, Rami Langar,* **Raouf Boutaba** *at University of Waterloo (Theme 3)*

Page for Notes

Chapter 4

Theme 4 Integrated Wireless/Optical Access

Research Team

Leslie Rusch (Theme Lead), Université of Laval
Tho LeNgoc, McGill University

4.1 Improving Throughput and Fairness in Virtualized 802.11 Networks through Association Control

Mahsa Derakhshani, Xiaowei Wang, **Tho Le-Ngoc**, McGill U., and **Alberto Leon-Garcia** University of Toronto

In virtualized 802.11 networks, it is challenging to provide service customization and fairness across multiple service providers, who share physical infrastructure and network capacity. This is mainly due to the use of a CSMA-based MAC, which couples flows of different virtual WLANs (as a result of unavoidable collisions). In a dense WLAN deployment, association control can facilitate a solution to control fairness and throughput among different ISPs. In this work, STA-AP association control is investigated and an optimization solution is formulated aiming to maximize overall network throughput while providing an airtime guarantee for each ISP. Subsequently, an algorithm is developed to reach the optimal solution by applying monomial approximation and using geometric programming iteratively.

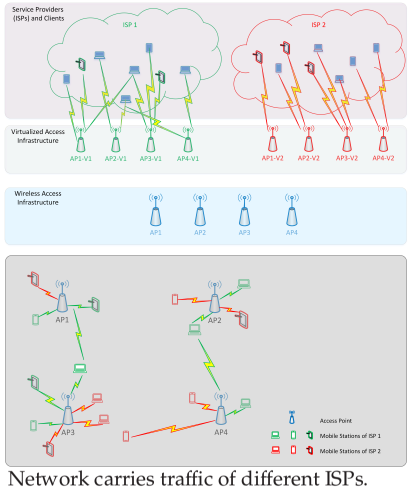
Improving Throughput and Fairness in Virtualized 802.11 Networks through Association Control

Mahsa Derakhshani, Xiaowei Wang, Tho Le-Ngoc, and Alberto Leon-Garcia

Abstract

In virtualized 802.11 networks, it is challenging to provide service customization and fairness across multiple service providers, who share physical infrastructure and network capacity. This is mainly because of using CSMA-based MAC which couples flows of different virtual WLANs due to unavoidable collisions. In a dense WLAN deployment, association control can facilitate a solution to control fairness and throughput among different Internet Service Providers(ISPs). In this work, Mobile Station(MS)-Access Point(AP) association control is investigated and an optimization problem is formulated aiming to maximize overall network throughput, while providing airtime guarantee for each ISP. Subsequently, an algorithm is developed to reach the optimal solution, by applying monomial approximation and using geometric programming iteratively.

System Model



- Network carries traffic of different ISPs.
- Each MS can be associated with several APs.
- τ_s^a: probability that MS s attempts to transmit at AP a in a generic time-slot.
- $x_s^a = \tau_s^a/(1-\tau_s^a)$: mean number of consecutive transmission attempts by MS s at AP a.
- $P_{\text{idle}}^a = \prod_{s\in\mathcal{S}}(1-\tau_s^a) = 1/(\prod_{s\in\mathcal{S}}(1+x_s^a))$

Throughput and Airtime Analysis

- Throughput of MS s at AP a is

$$T_s^a = \frac{P_{\text{succ},s}^a T_{\text{TXOP}}}{P_{\text{idle}}^a \sigma + (1-P_{\text{idle}}^a)T}$$

- Probability of successful transmission

$$P_{\text{succ},s}^a = \tau_s^a \prod_{s'\neq s}(1-\tau_{s'}^a) = \frac{x_s^a}{\prod_{s'\in\mathcal{S}}(1+x_{s'}^a)}$$

- Consequently,

$$T_s^a = \frac{x_s^a t}{\prod_{s'\in\mathcal{S}}(1+x_{s'}^a) - t'}$$

- Airtime of MS s at AP a is

$$T_{\text{air},s}^a = \frac{P_{\text{coll},s}^a T + P_{\text{succ},s}^a T_{\text{TXOP}}}{P_{\text{idle}}^a \sigma + (1-P_{\text{idle}}^a)T}$$

- Probability of collision

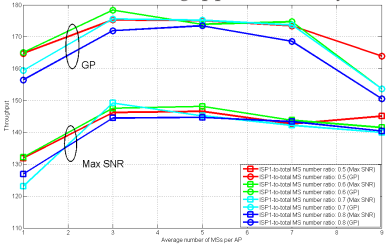
$$P_{\text{coll},s}^a = \tau_s^a[1-\prod_{s'\neq s}(1-\tau_{s'}^a)] = \frac{x_s^a[\prod_{s'\neq s}(1+x_{s'}^a)-1]}{\prod_{s'\in\mathcal{S}}(1+x_{s'}^a)}$$

- Consequently,

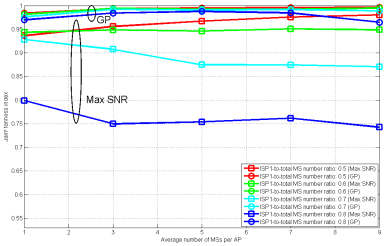
$$T_{\text{air},s}^a = \frac{x_s^a \prod_{s'\neq s}(1+x_{s'}^a)}{\prod_{s'\in\mathcal{S}}(1+x_{s'}^a) - t'}$$

Numerical Results

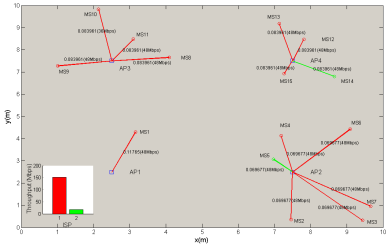
WLAN Throughput vs Density



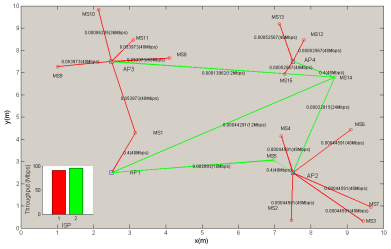
ISP Fairness vs Density



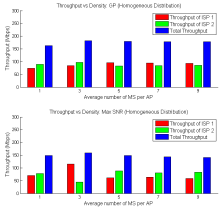
Association Result: Max SNR



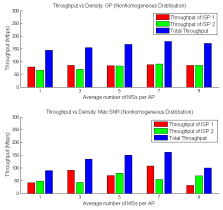
Association Result: GP



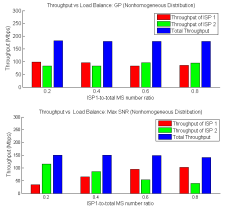
Homogeneous Distribution



Non-homogeneous Distribution



Varying Load Balance



Association Control

- MS-AP association optimization problem

$$\max_{\boldsymbol{X}} \sum_{s\in\mathcal{S},a\in\mathcal{A}} \frac{x_s^a r_s^a t}{\prod_{s'\in\mathcal{S}}(1+x_{s'}^a) - t'}, \text{ s.t.,} \quad (1)$$

$$\text{C11}: \sum_{s\in\mathcal{S}_i,a\in\mathcal{A}} \frac{x_s^a \prod_{s'\neq s}(1+x_{s'}^a)}{\prod_{s'\in\mathcal{S}}(1+x_{s'}^a) - t'} \geq \eta_i, \forall i \in \mathcal{I}$$

$$\text{C12}: 0 \leq x_s^a \leq \eta_x, \ \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

- MS-AP association through complementary geometric programming

$$\min_{\boldsymbol{X},\boldsymbol{Y},\boldsymbol{T},x_0} \ x_0, \ \text{s.t., C12,} \quad (2)$$

$$\text{C21}: \frac{M}{x_0 + \prod_{s\in\mathcal{S},a\in\mathcal{A}}\left(\frac{x_s^a r_s^a t}{y^a}\right)} \leq 1$$

$$\text{C22}: \frac{\prod_{s\in\mathcal{S}}(1+x_s^a)}{t'+y^a} = 1, \ \forall a \in \mathcal{A}$$

$$\text{C23}: \frac{\eta_i + 1}{1 + \sum_{s\in\mathcal{S}_i,a\in\mathcal{A}} \frac{x_s^a \prod_{s'\neq s} t_s^a}{y^a}} \leq 1, \ \forall i \in \mathcal{I}$$

$$\text{C24}: \frac{t_s^a}{1+x_s^a} = 1, \ \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

Conclusion

It is shown that proposed MS-AP association control algorithm ensures fairness among competing ISPs regardless of network conditions, such as number of MSs per ISP and MS distribution. Furthermore, despite minimum airtime reservation for each ISP, proposed algorithm improves total network throughput comparing to the Max-SNR algorithm.

References

[1] A. Checco and D. J. Leith. Fair virtualization of 802.11 networks. *IEEE/ACM Trans. Netw.*

[2] D. Gong and Y. Yang. On-line ap association algorithms for 802.11 n wlans with heterogeneous clients. *IEEE Trans. Comput.*, Aug. 2013.

Acknowledgment

This work is funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP394424-10.

FIGURE 4.1: *Improving Throughput and Fairness in Virtualized 802.11 Networks through Association Control from Mahsa Derakhshani, Xiaowei Wang,* ***Tho Le-Ngoc*** *at McGill University and* ***Alberto Leon-Garcia*** *at University of Toronto (Theme 4)*

4.2 QoS-Oriented Slice Provisioning in Wireless Virtualized Networks

Vikas Jumba, Saeideh Parsaei Fard, Mahsa Derakhshani, **Tho Le-Ngoc**, McGill U.

Wireless network virtualization is a promising paradigm to increase the spectrum efficiency via allocating bundles of physical-layer resources (e.g., power and spectrum) to different service providers, referred to as slices. In this work, to efficiently utilize resources between slices, we propose a resource allocation algorithm by maximizing the total throughput of a virtualized wireless network (VWN), while preserving a minimum required quality of service (QoS) for each slice. Due to channel variations, VWN always encounters an outage probability, i.e., the QoS does not hold. To prevent this issue, we present a dynamic admission control algorithm. Via simulation results, we demonstrate the performance of our proposed algorithms.

QoS-Oriented Slice Provisioning in Wireless Virtualized Networks

Vikas Jumba,Saeideh Parsaei Fard,Mahsa Derakhshani and Tho Le-Ngoc

Abstract

Wireless network virtualization is a promising paradigm to increase the spectrum efficiency via allocating bundles of physical layer resources (e.g., power and spectrum) to different service providers, referred to as slices [1, 2, 3]. In this work, to efficiently utilize resources between slices, we propose a resource allocation algorithm by maximizing the total throughput of virtualized wireless network (VWN), while preserving a minimum required quality of service (QoS) for each slice. Due to the channel variations, VWN always encounters outage probability, i.e., the QoS does not hold. To prevent this issue, we present a dynamic admission control algorithm. Via simulation results, we demonstrate the performance of our proposed algorithms.

System Model

We consider the VWN where one base station (BS) shares the bundles of resources between two groups of slices:
1) **Resource-based Slices**: minimum number of subcarriers, K_g^{rsv}, is preserved for each slice
2) **Bandwidth-based Slices**: minimum required bandwidth, R_g^{rsv}, is guaranteed for each slice

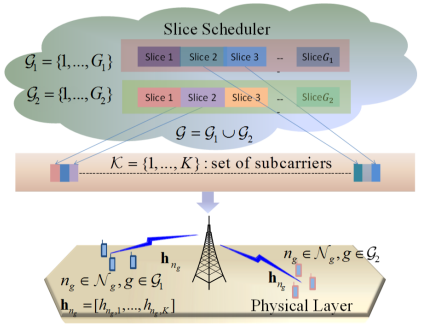


Figure 1: System model of VWN

- Transmit power vector for all users: $\mathbf{P} = \{\mathbf{P}_1, ..., \mathbf{P}_G\}$ where $\mathbf{P}_g = \{\mathbf{P}_{n_1}, ..., \mathbf{P}_{n_G}\}$ and $\mathbf{P}_{n_g} = \{P_{n_g,1}, ..., P_{n_g,K}\}$ for all n_g.
- Subcarrier assignment vector for all users: $\mathbf{w} = \{\mathbf{w}_1, ..., \mathbf{w}_G\}$ where $\mathbf{w}_g = \{\mathbf{w}_{n_1}, ..., \mathbf{w}_{n_G}\}$ and $\mathbf{w}_{n_g} = \{w_1, ..., w_K\}$.

Slice Provisioning and its Performance

The resource provisioning algorithm is defined based on the optimization problem with the objective to maximize the total rate of VWN subject to the minimum requirement of each slice.
For simulation results, a single BS virtualized with two slices g_1 and g_2 and $K = 124$ is considered. For Figs. 2 and 4, we set four users in each slice, and for Fig. 3, we have $P_{n_g,\max}/\sigma = 20$ dB.

$$\max_{\mathbf{w},\mathbf{P}} \sum_{g\in\mathcal{G}} \sum_{n_g\in\mathcal{N}_g} \sum_{k\in\mathcal{K}} w_{n_g,k} \log(1 + \frac{P_{n_g,k} h_{n_g,k}}{\sigma^2}),$$

subject to

$$\text{C11: } \sum_{n_g\in\mathcal{N}_g} \sum_{k\in\mathcal{K}} w_{n_g,k} \log(1 + \frac{P_{n_g,k} h_{n_g,k}}{\sigma^2}) \geq R_g^{\text{rsv}}, \ \forall g \in \mathcal{G}_1,$$

$$\text{C12: } \sum_{n_g\in\mathcal{N}_g} \sum_{k\in\mathcal{K}} w_{n_g,k} > K_g^{\text{rsv}}, \ \forall g \in \mathcal{G}_2,$$

$$\text{C13: } \sum_{g\in\mathcal{G}} \sum_{n_g\in\mathcal{N}_g} w_{n_g,k} \leq 1, \ w_{n_g,k} \in [0,1],$$

$$\text{C14: } \sum_{k\in\mathcal{K}} w_{n_g,k} P_{n_g,k} \leq P_{n_g,\max}, \ \forall n_g \in \mathcal{N}_g, \forall g \in \mathcal{G}.$$

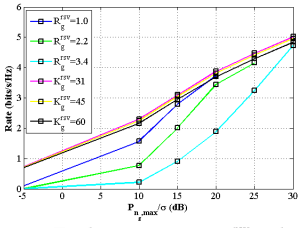


Figure 2: Total rate vs $P_{n_g,\max}$, R_g^{rsv} and K_g^{rsv}.

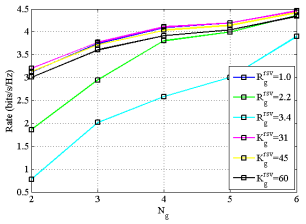


Figure 3: Total rate vs number of users per slice, R_g^{rsv} and K_g^{rsv}.

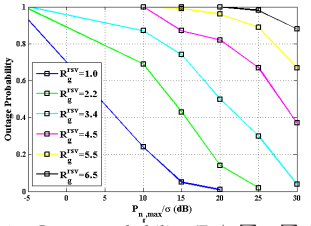


Figure 4: Outage probability ($P_r(\sum_{n_g\in\mathcal{N}_g}\sum_{k\in\mathcal{K}} R_{n_g,k} \leq R_g^{\text{rsv}})$) vs $P_{n_g,\max}$ and R_g^{rsv}.

Admission Control Policy and its Performance

Due to channel fading, maximum power limitations and user's mobility, there always exists a chance of outage probability, when holding the minimum required rate is not feasible, as depicted in Fig. 4. To deal with this issue, we introduced an admission control algorithm where the required QoS of each slice is adjusted based on channel gains. We consider $P_{n_g,k} = x_{n_g,k}/w_{n_g,k}$ for the admission control policy.

$$\max_{\mathbf{x},\mathbf{w},\epsilon} \sum_{g\in\mathcal{G}} \sum_{n_g\in\mathcal{N}_g} \sum_{k\in\mathcal{K}} \log(1 + \frac{x_{n_g,k} h_{n_g,k}}{\sigma w_{n_g,k}}) - \alpha_g \epsilon_g,$$

subject to C13-C14,

$$\text{C21: } \sum_{n_g\in\mathcal{N}_g} \sum_{k\in\mathcal{K}} \log(1 + \frac{x_{n_g,k} h_{n_g,k}}{\sigma w_{n_g,k}}) \geq R_g^{\text{rsv}} - \epsilon_g, \ \forall g \in \mathcal{G}_1$$

$$\text{C22: } \sum_{n_g\in\mathcal{N}_g} \sum_{k\in\mathcal{K}} w_{n_g,k} \geq K_g^{\text{rsv}} - \epsilon_g, \ \forall g \in \mathcal{G}_2.$$

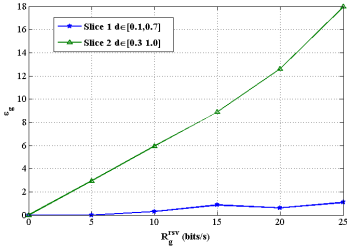


Figure 5: ϵ_g vs R_g^{rsv} and distances of users from BS

Algorithm

- **Initialization**: Choose feasible values of $\mathbf{w}^*(l = 0)$ and $\mathbf{P}^*(l = 0)$, and set $l = 1$.
- **Step 1**: For fixed $\mathbf{P}^*(l-1)$, solve subcarrier assignment subproblem to derive $\mathbf{w}^*(l)$.
- **Step 2**: For fixed $\mathbf{w}^*(l)$, solve power allocation problem to derive $\mathbf{P}^*(l)$.
- When $||(\mathbf{P}(l) - \mathbf{P}(l-1))|| \leq \epsilon$, where $\epsilon = 0.001$, stop. Otherwise, $l = l + 1$ and goto **Step 1**.

Discussion

- Fig. 2 demonstrates that increasing power increases the total rate due to expanding the feasibility set.
- Fig. 3 shows the user diversity gain in VWN.
- Fig. 4 emphasis on the importance of admission control.
- Fig. 5 demonstrates the performance of admission control vs the location of users and R_g^{rsv}.

Conclusion

- We proposed an iterative algorithm for subcarrier and power allocation in OFDMA-based VWN with minimum bandwidth or subcarriers guaranteed for slice owners.
- An admission control policy is proposed for the system model to deal with the outage probability based on the channel gains of end users.
- We are planning to extend this system model for multiple BS while considering the maximum delay for users and their queue stability.

References

[1] P. Lv, X. Wang, Y. Yang, and M. Xu, "Network virtualization for smart grid communications," *IEEE Systems Journal*, vol. 8, no. 2, pp. 471–482, June 2014.

[2] F. Fu and U. Kozat, "Stochastic game for wireless network virtualization," *IEEE/ACM Transactions on sNetworking*, vol. 21, no. 1, pp. 84–97, Feb 2013.

[3] H. Wen, P. Tiwary, and T. Le-Ngoc, "Current trends and perspectives in wireless virtualization," in *International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT) 2013*, Aug 2013, pp. 62–67.

Acknowledgment

This work is funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP394424-10.

FIGURE 4.2: *QoS-Oriented Slice Provisioning in Wireless Virtualized Networks from Vikas Jumba, Saeideh Parsaei Fard, Mahsa Derakhshani, **Tho Le-Ngoc** at McGill University (Theme 4)*

4.3 Management of Virtualized and Software-Defined Wireless Infrastructures: Issues, Requirements, Framework and Specifications

Prabhat Kumar Tiwary, Quang-Dung Ho, **Tho Le-Ngoc**, McGill U.

Virtualization of wireless infrastructures furnishes a rich variety of abstracted and sharable wireless resources encompassing different wireless technologies and standards. On the other hand, the service-oriented architecture model is affecting the tenant ecosystem by allowing tenants to own, share, borrow and manage the virtual wireless resources in many desirable ways. The setup and management of such diverse virtual resources is non-trivial. Moreover, supporting a service-oriented architecture and software-defined paradigm adds additional management complexity. In this regard, this poster first discusses the main existing and foreshadowed management issues. Next, the management requirements for a virtualized and software-defined wireless infrastructure are explored. In addition, a draft initial management framework (along with a set of management specifications) has been put forward. The framework and the specifications have been prototyped in Aurora (as Aurora-Manager and Aurora-Tenant), which is demonstrated through a flow-based application scenario.

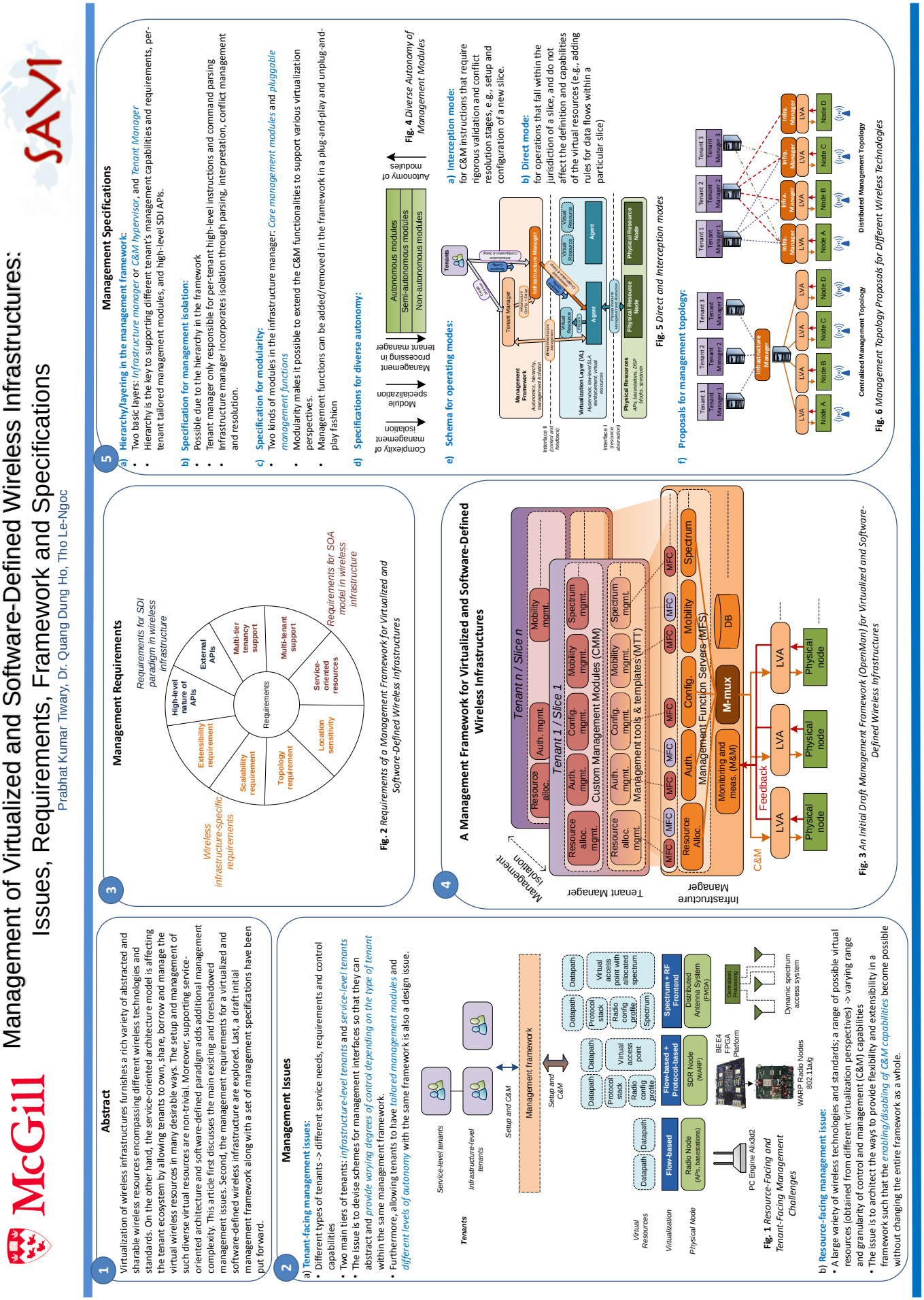


FIGURE 4.3: *Management of Virtualized and Software-Defined Wireless Infrastructures: Issues, Requirements, Framework and Specifications from Prabhat Kumar Tiwary, Quang-Dung Ho,* ***Tho Le-Ngoc*** *at McGill University (Theme 4)*

4.4 Aurora: A Virtualization and Software-Defined Infrastructure Framework for Wireless Networks

Prabhat Kumar Tiwary, Kevin Han, Hoai-Phuoc Truong, Quang-Dung Ho, **Tho Le-Ngoc**, McGill U.

This poster introduces Aurora, a virtualization framework and testbed platform for supporting multiple types of virtualization techniques and architectures specifically applied to wireless technologies. After a brief overview of the wireless virtualization perspectives, the Aurora resource abstraction model is discussed, and the software architecture and the design principles behind Aurora are explained. Next, the current deployment of Aurora within SAVI is outlined and the prospects of Aurora in supporting different virtualization perspectives and potential ways to extend to different wireless technologies are sketched.

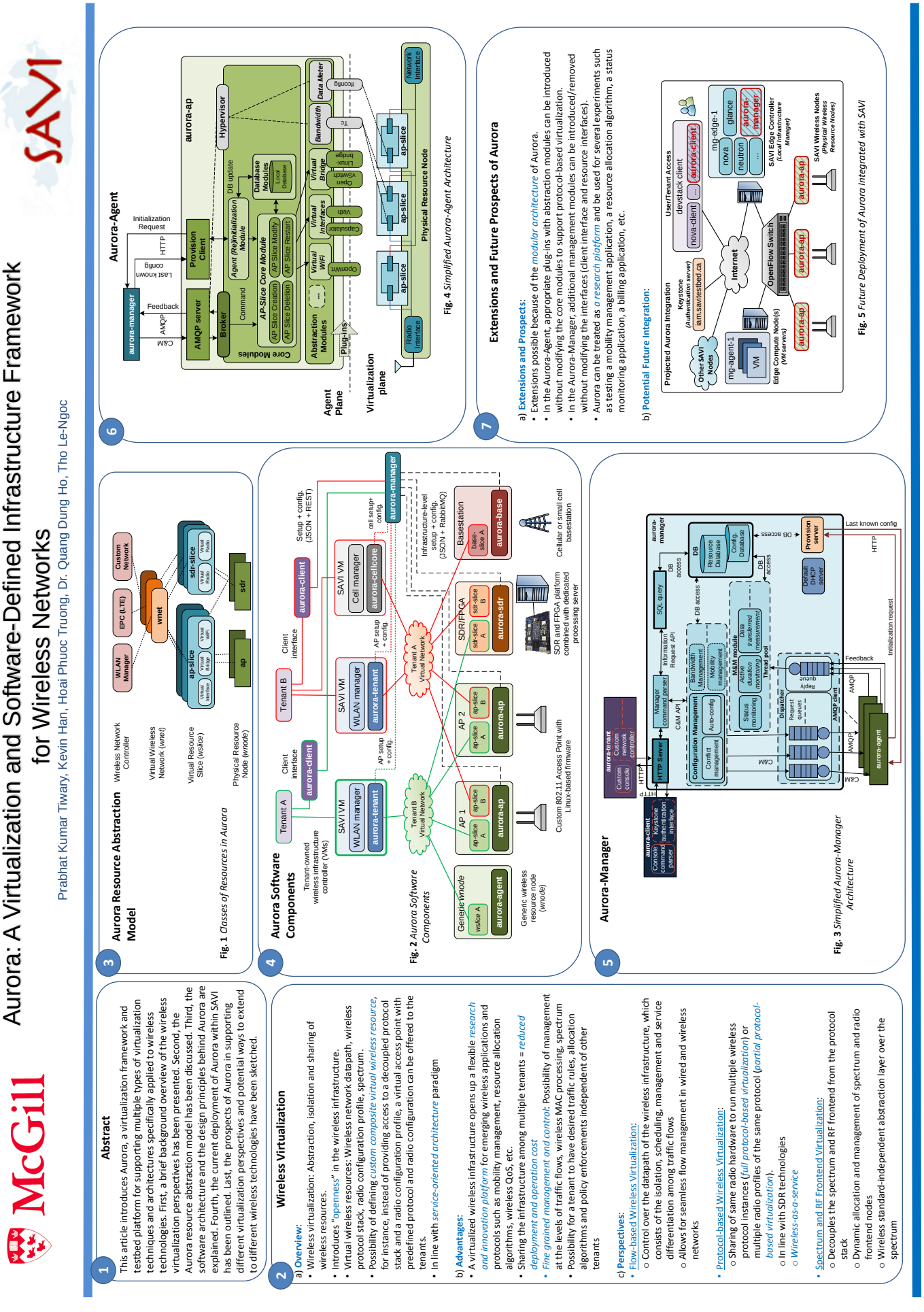


FIGURE 4.4: *Aurora: A Virtualization and Software-Defined Infrastructure Framework for Wireless Networks from Prabhat Kumar Tiwary, Kevin Han, Hoai-Phuoc Truong, Quang-Dung Ho,* ***Tho Le-Ngoc*** *at McGill University (Theme 4)*

4.5 Full-duplex WiFi Analog Signal Transmission with Digital Signal Downlink in Radio-over-Fiber System Employing RSOA-based WDM-PON Architecture

Truong An Nguyen, **Leslie A. Rusch**, U. Laval

We experimentally demonstrate bidirectional analog WiFi signal transmission in a digital wavelength division multiplexing passive optical network system employing reflective semiconductor optical amplifier (RSOA). A downlink (DL) signal comprising a 1 Gb/s On-Off Keying (OOK) and a WiFi signal is transmitted simultaneously with an uplink (UL) WiFi signal. The DL optical carrier is reused by the RSOA for UL transmission. We transmit pass-band WiFi signals at 2.4 GHz band without frequency translation even though the RSOA's response is limited at 1.2 GHz. At the bit-error-rate (BER) threshold of 2.10-3 before forward error correction (FEC), we achieve the transmission link up to 20 km (64-QAM), 30 km (16-QAM), and 40 km (QPSK). In all cases, the 1 Gb/s digital downlink signal is always error-free.

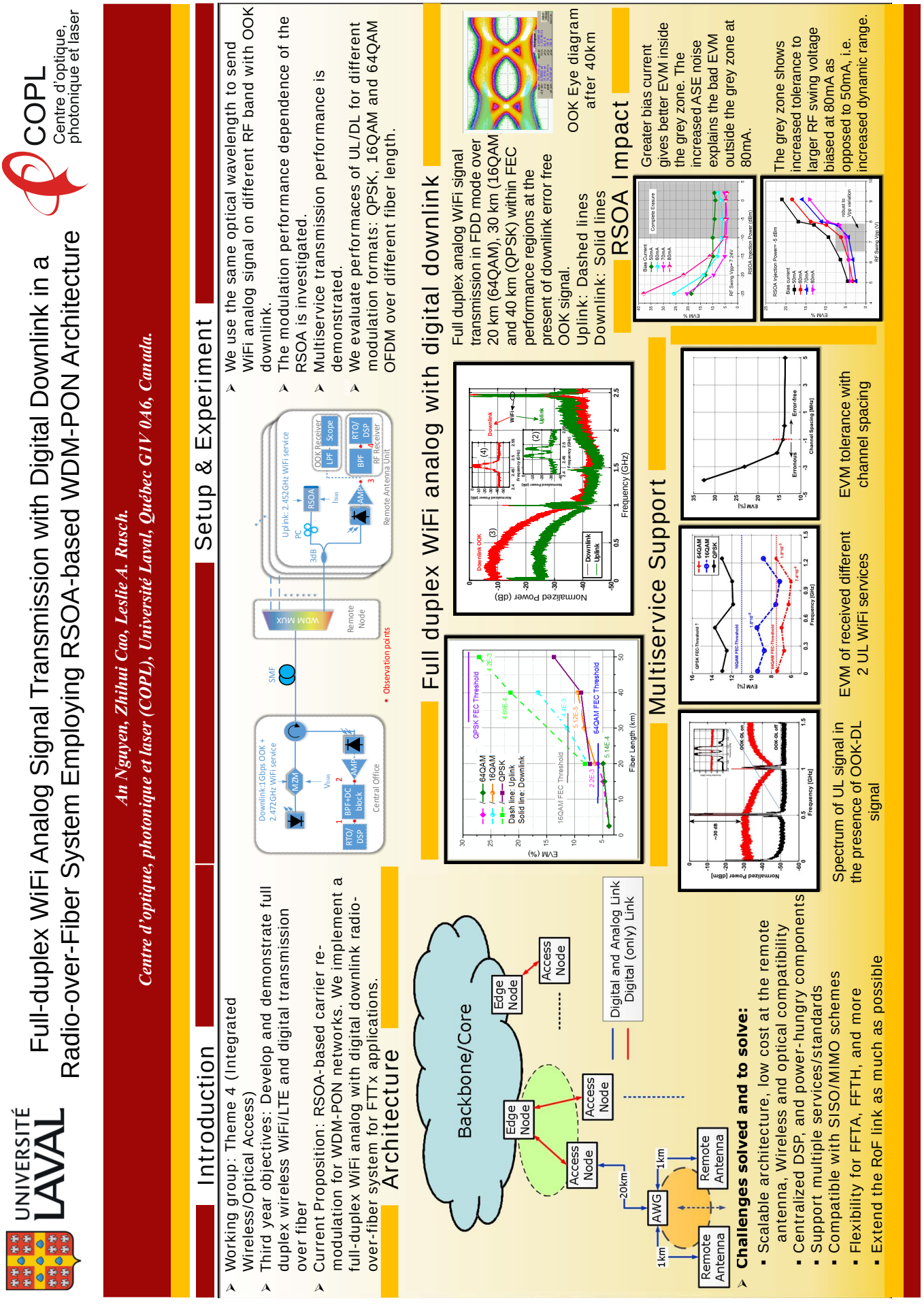


FIGURE 4.5: *Full-duplex WiFi Analog Signal Transmission with Digital Signal Downlink in Radio-over-Fiber System Employing RSOA-based WDM-PON Architecture from Truong An Nguyen and* ***Leslie A. Rusch*** *at Université of Laval (Theme 4)*

4.6 An Application of Aurora with 802.11 wireless networks and Flow-based Virtualization

Hoai-Phuoc Truong, Kevin Han, Prabhat Kumar Tiwary, Quang-Dung Ho, **Tho Le-Ngoc**, McGill U.

This demonstration presents an application of the Aurora framework in a healthcare setting using the IEEE 802.11 wireless technology and flow-based virtualization. A centralized management topology is used with the Aurora-Manager running on an Ubuntu 12.04 laptop and Aurora-Agents on the PC engines. A key feature demonstrated is flexible ap-slice setup and management using Aurora APIs through Aurora-Client dashboard. It is shown that ap-slices can share the same physical AP. Similarly, the advantage of Aurora resource abstraction model (specifically wnet) in joint management of ap-slices is showcased. With regard to this scenario, some relevant management statistics such as the status of a slice for the purpose of billing are also shown. Finally, the internal management steps such as environment variable setup, API calls, parsing, conflict management, resolution, database queries, message queuing, AP polling, status reporting, database updates and response generation for the user (tenant) while execution of an Aurora command is explained with the help of a timeline diagram.

An Application of Aurora with 802.11 Wireless Networks and Flow-based Virtualization

Kevin Han, Hoai Phuoc Truong, Prabhat Kumar Tiwary, Quang-Dung Ho, and Tho Le-Ngoc
Broadband Communications Research Laboratory, McGill University

McGill

SAVI

Abstract

This demonstration presents a real-life application of the Aurora framework in a healthcare-related scenario. The IEEE 802.11/WiFi radio technology implemented with flow-based virtualization capability is illustrated. A centralized management topology is used with the Aurora Manager running on a PC and Aurora Agents running on the WiFi Access Points (APs). A key feature demonstrated is flexible ap-slice setup and management using Aurora APIs through Aurora Client dashboard. It can be seen that multiple ap-slices can share the same physical AP. Additionally, the advantage of Aurora resource abstraction model (specifically wnet) in joint management of ap-slices is showcased. With regard to this scenario, a number of management statistics of wireless resource slices for the purposes of wireless network administration and billing are also illustrated. Furthermore, management steps such as AP polling, status reporting, database updates, etc., while execution of an Aurora command are explained with a timeline diagram.

The Demonstration Scenario

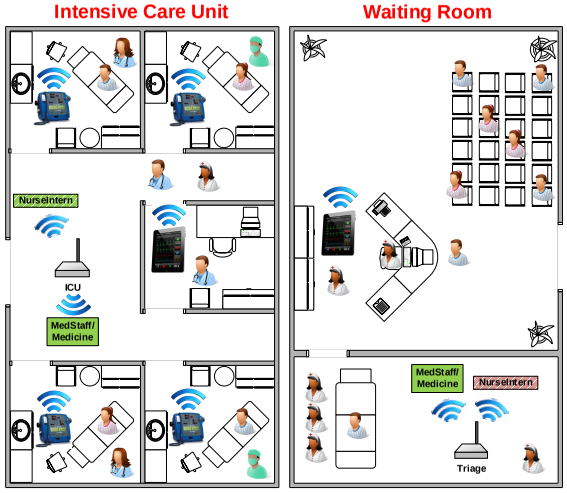


Fig. 1. Scenario setup in a hospital emergency department

Aurora Component Architecture

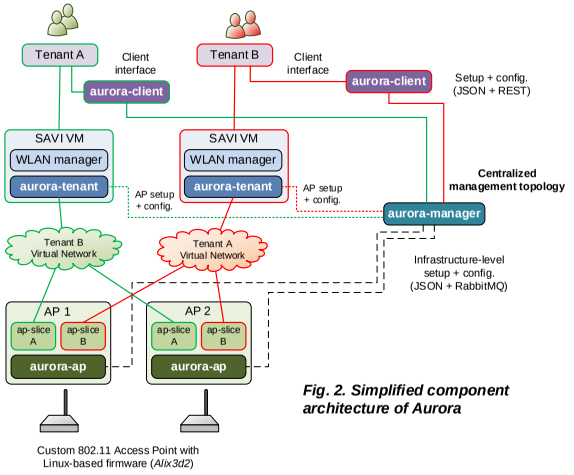


Fig. 2. Simplified component architecture of Aurora

Flexible and Dynamic Setup and Management of a WLAN with Aurora

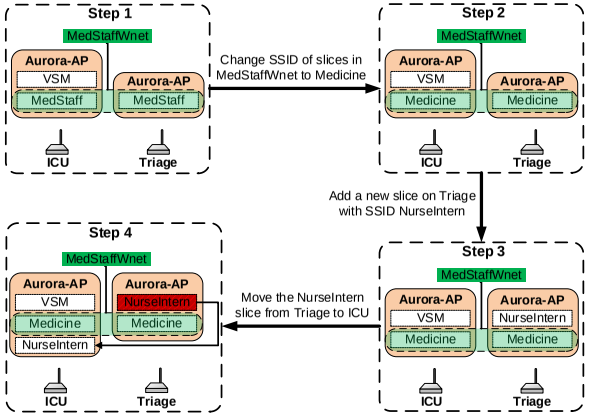


Fig. 3. Setup and management of virtual wireless resources with Aurora in a health care-related scenario

Aurora Management Flow

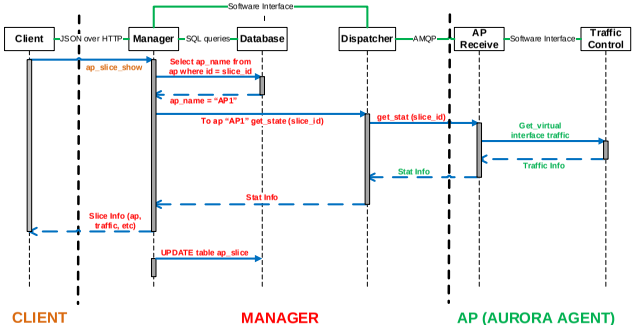


Fig . 4. A representation of internal management calls and responses pertaining to ap-slice-show API

States and Life Cycle of a Slice

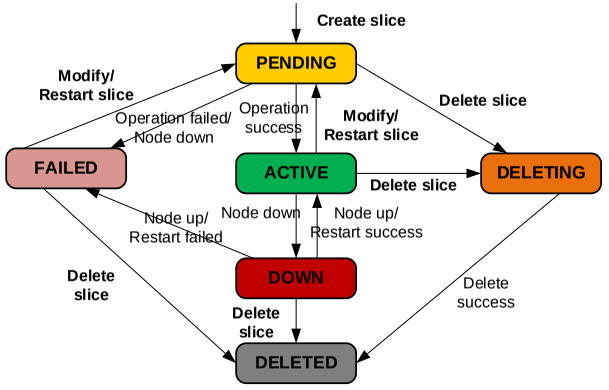


Fig. 5. Different Slice States in Aurora Architecture.

Fig. 6. The Life Cycle of the "NurseIntern" Slice

FIGURE 4.6: *An Application of Aurora with 802.11 wireless networks and Flow-based Virtualization from Hoai-Phuoc Truong, Kevin Han, Prabhat Kumar Tiwary, Quang-Dung Ho,* ***Tho Le-Ngoc*** *at McGill University (Theme 4)*

Page for Notes

Chapter 5

Theme 5 SAVI Application Platform Testbed

Research Team

Alberto Leon-Garcia (Theme Lead), University of Toronto
Raouf Boutaba, University of Waterloo
Paul Chow, University of Toronto
Yashar Ganjali, University of Toronto
Marin Litoiu, York University
Greg Steffan, University of Toronto (Posthumous)
Sudhakar Ganti, University of Victoria
Hadi Bannazadeh, SAVI Architect

5.1 Virtualized Reconfigurable Hardware in the SAVI Testbed

Stuart Byma, **Hadi Bannazadeh, J. Gregory Steffan, Alberto Leon-Garcia, Paul Chow**, U. Toronto

This poster presents a novel approach for integrating virtualized FPGA-based hardware accelerators into the SAVI testbed, with minimal virtualization overhead. Partially reconfigurable regions inside FPGAs are offered as generic cloud resources through the testbed OpenStack management system, thereby allowing users to "boot" custom designed or predefined network-connected hardware accelerators with the same commands they would use to boot a regular Virtual Machine. The poster details the hardware and software framework to enable this virtualization, and also shows two example use cases – a custom UDP load balancer and an application used to give OpenFlow the ability to execute matching and actions on VXLAN tunneled traffic. The latter application will be showcased in a demonstration.

Virtualized Reconfigurable Hardware in the SAVI Testbed

Stuart Byma, J. Gregory Steffan, Hadi Bannazadeh, Alberto Leon-Garcia, Paul Chow

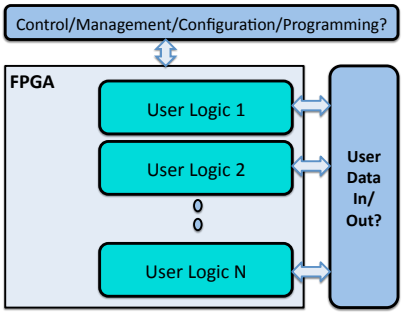
Department of Electrical and Computer Engineering
University of Toronto, Ontario, Canada

Motivation

- SAVI Testbed contains FPGAs
- These FPGAs are not virtualized – not sharable, flexible, or scalable
- Goal: virtualize FPGA resources to enable cloud-based hardware acceleration

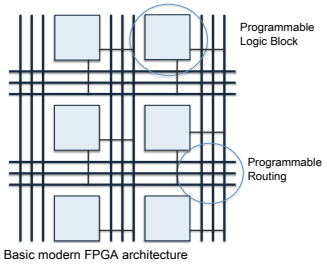
What is Virtualization?

- Adding an abstraction layer
  - e.g. PC hardware virtualization
  - Abstracts physical hardware, allowing "sharing" of hardware by several virtual computers
- Goal: Virtualize FPGAs
  - Allow multiple users to "share" the same FPGA fabric
  - Enable easy, flexible programmability
  - Make FPGA accelerators appear as cloud resources - create and tear down on the fly
- Challenge - How to accomplish this?



Wait, What is an FPGA?

- Field Programmable Gate Array
  - Tiled array of programmable logic blocks, interconnected by programmable routing architecture
  - Hard block RAMs, DSP Blocks (multipliers), high speed transeivers
  - Result: Fully reconfigurable hardware operating at speeds >500 MHz
- Drawback - Requires complex CAD flow to map design to the device
  - Big learning curve to implement significant designs

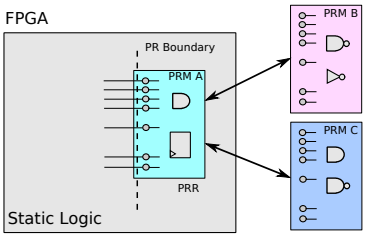


Acknowledgements

XILINX
SAVI
CMC MICROSYSTEMS
NSERC CRSNG

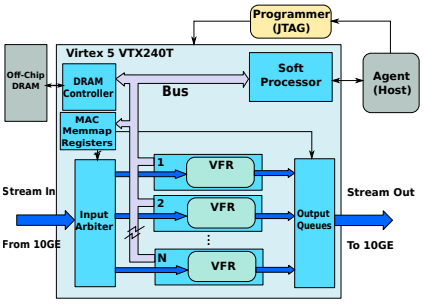
Hardware Virtualization using Partial Reconfiguration

- Adding an abstraction layer
- Split the FPGA into multiple **Virtualized FPGA Resources** using PR [1]
- Partial Reconfiguration – the ability to reconfigure a section of an FPGA
  - At runtime
  - Without affecting other running circuits



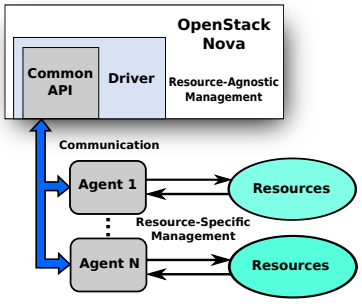
SAVI Testbed Prototype System

- MicroBlaze for command and control
- Four VFRs, 10G network-attached accelerators
- Each VFR has 11376 LUTs and 19 Block RAMs
- Script-based automated compile system



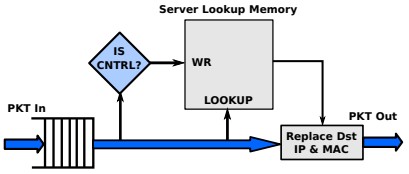
OpenStack Integration

- Make VFRs available as resources in the SAVI testbed
  - Through a *Driver-Agent* abstraction



Application: Load Balancer

- Round-robin scheduling of UDP packets to servers
- Created using Vivado High-Level Synthesis, run inside VFR system
- Control packets to add servers to scheduler



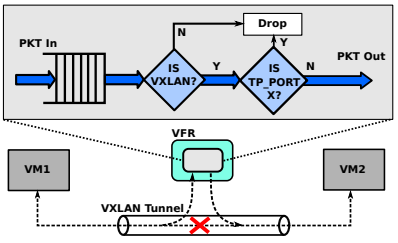
Results

- Comparison to VM-based load balancer in Python
- Measure latency while flooding at variable rates
- Much higher throughput!

VM	VFR
45 MB/s	>100 MB/s

Application: Enhancing OpenFlow Capabilities

- OpenFlow has a limited match and action set
  - Goal: Use virtualized hardware to enhance these
- Create VFR hardware to match and drop VXLAN encapsulated packets
  - Based on Transport Layer port
- Switch flows to redirect VXLAN traffic to VFR



Results

- VXLAN Tunnel between two VMs
- `iperf` with and without flow redirection to VFR
- Virtually no difference in throughput!

Without FWDing	With FWDing
517.4 Mb/s	513.2 Mb/s

References

[1] S. Byma, J. G. Steffan, H. Bannazadeh, A. Leon-Garcia, and P. Chow.
FPGAs in the Cloud: Booting Virtualized Hardware Accelerators with OpenStack.
In *22nd Internation Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2014.

bymastua@eecg.toronto.edu

FIGURE 5.1: *Virtualized Reconfigurable Hardware in the SAVI Testbed from Stuart Byma,* ***Hadi Bannazadeh, J. Gregory Steffan, Alberto Leon-Garcia, Paul Chow*** *at University of Toronto (Theme 5)*

5.2 Real-time Enhancement of IMS Quality of Service using SAVI SDI

Lilin Zhang, Jieyu Lin, **Hadi Bannazadeh, Alberto Leon-Garcia**, U. Toronto

Traditional IMS/VoIP services lack real-time bandwidth provisioning and/or reconfiguration to improve the VoIP quality. In this work, we use SAVI SDI to prototype a novel method of enhancing VoIP quality by using its converged and centralized manner of resource management. We will demonstrate that with a reconfiguration operation from the SDI platform, a bandwidth thirsty HD video call can be elevated to a higher QoS in real time to achieve visibly significant quality improvement compared to a call made over best effort VoIP connection.

Real-time Enhancement of VoIP Media Quality using SAVI SDI

Lilin Zhang, Jieyu Lin, Hadi Bannazadeh and Alberto Leon-Garcia
ECE Department, University of Toronto

Abstract

Traditional IMS/VoIP services lack real-time network provisioning and/or reconfiguration to improve the VoIP quality. In this work, we use SAVI SDI to prototype a novel method of enhancing VoIP media quality by using its converged and centralized manner of resource management. We will demonstrate that with a reconfiguration operation from the SDI platform, the media quality of a bandwidth thirsty HD video call can be elevated to a higher level in real time to achieve visibly significant quality improvement compared to a call made over best effort connection.

Introduction

VoIP service is not tolerant of network packet loss, and also requires large end-to-end bandwidth for HD media flows between end users. The problem gets worse quickly with a number of VoIP sessions, or file transfer, or Video streaming simultaneously on-going in the access network. As the access network resources are soon used up, it leads to longer queues/packet drops along the congested area, which significantly degrades the media quality experience of the end users affected.

Traditional IP networks do not support a fast re-configuration to resolve such congestion-lead VoIP media degration in a real-time manner. OpenFlow technology provides a solution:

1. Forwarding logic → *Controller*;
2. Forwarding processing → OpenFlow-enabled switches;
3. *Controller* ⟷ OF-enabled switches (in OpenFlow protocol).

SAVI testbed has provided the application platform for the VoIP service. It has:

- created the compute and storage pool by leveraging opensource virtualization technologies;
- also expanded with the *virtual network functions* to configure/reconfigure the OpenFlow-enabled networks.

In this work, **we demonstrate the capability of SAVI SDI platform** that with one network path reconfiguration from the platform, the media flow of an on-going video call session can be re-routed through a less congested path, and the end users are able to see visibly significant quality improvement in media fluidity of the call session.

Determinants of Media Quality

The media quality of VoIP call sessions is determined by:

1. Video codecs → efficiency, bitrate → lossiness, image capture size → pixels, and frame rate → fluidity.
2. Network condition - In particular, VoIP is not tolerant of large packet loss. See more in Table 1
3. User client - as the real-time video/audio are captured, compressed, transcoded and transmitted on the fly, the more capable the user client, the better the media quality.

Fig. 1 shows the side-by-side snapshots of the video call encoded in low bitrate and the one encoded in high bitrate.

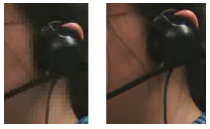


Figure 1: Bitrate = $297Kbps$ (left) Vs. Bitrate = $2Mbps$ (right), both captured by Logitech c930e 1080p webcam ("p" means progressive scanning, "1080p" means that the webcam is capable of scanning up to 1600×1080 pixels every $\frac{1}{30}$ second)

Table 1 highlights the network requirements for VoIP services. Generally for audio codec G.711, 1% packet loss can"significantly degrade" a VoIP call. Other codecs can tolerate even less packet loss. For instance, the default G.729 codec requires packet loss far less than 1% to avoid audible errors. Video codec has similar stringent requirements in low pakcet loss.

QoS	Requirement
Latency	≤ 150 ms one-way [1]
Packet loss	$\leq 1\%$
Bandwidth	$\geq$ datarate

Table 1: VoIP QoS Requirements

Table 2 lists the datarates of video calls of various resolution sizes, that as the image capture size increases, the average datarate also increases. As pointed out in Table 1, the more pixels were captured by the webcam, the more bandwidth will be required for the end-to-end media flow.

Size	**Avg. datarate** $(1.0e+07)$	σ $(1.0e+06)$
176×144	0.0340 bps	0.0090
352×288	0.1086 bps	0.0219
640×480	0.2917 bps	0.0442
1280×720	0.4824 bps	0.1608
176×144	0.5955 bps	0.1055
352×288	1.6173 bps	0.6679
640×480	2.7497 bps	0.9270
1280×720	2.5734 bps	2.0420

Table 2: Various datarate for different capture sizes encoded in 297 Kbps (top) and encoded in 2 Mbps (bottom)

IMS Network Embedding

The IMS network consists of two parts:

- *Signalling plane* - The flows of SIP [2] messages sent between UE ⟷ IMS domain;
- *Media plane* - The flows of actual audio/video RTP packets sent between UE ⟷ UE.

In Fig. 2, the logical IMS network (top) is mapped to the physical switches and routers in the access network (bottom).

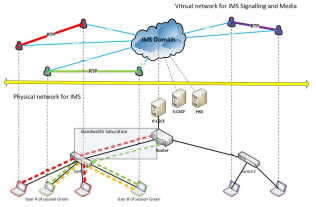


Figure 2: Virtual Network Mapping for IMS service

The **default forwarding path** of the GREEN session (the green dash line) leads to potential *network congestion* as it uses the shared link with the RED session, and consequently downgrades the media quality for both RED session and GREEN session. Its **alternative forwarding path** (orange dash line) could enhance the media quality for both call sessions.

VoIP Service on SAVI Testbed

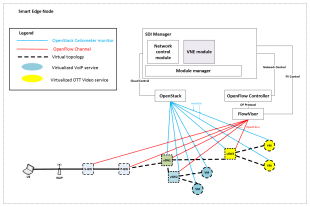


Figure 3: Virtualizing Multimedia Services in SAVI testbed

SAVI testbed leverages OpenStack to provide compute and storage pool and measurement for applications:

- It monitors the usage statistics of each physical and virutal servers and switches (blue lines in Fig. 3).

This feature could be made use of to locate the "hotspot" in the network.

SAVI testbed has converged the configuration of network resources on to the platform as well:

- The centralized controller communicates in OpenFlow protocol (red lines in Fig. 3) to each and every switch, including the soft switches and physical Pronto OpenFlow switches.

Enhancement Achieved using SAVI SDI

The topology of our demo is shown in Fig. 2. We use a pair of traffic nodes (generating up to 928Mbps), to simulate bandwidth-hungery applications, such as simultaneous video call sessions, or file transfers, or video streaming.

We use two computers (belonging to different subnets) to simulate the GREEN session, making a point-to-point video call. The visible difference brought by the SDI enhancement is the **fluidity** of the video call session - the media freezes (using the default path) and media restores (using the alternative path) in presence of the traffic congestion brought out by the RED session.

The media flow is uniquely identified:

- < 17 *(protocol type for UDP), ip.src, ip.dst, eth.src, eth.dst* $>$

To enhance the media quality - bypassing the congested area, the centralized *Controller* issues the OpenFlow actions to Switch 1 (*manually specified*):

- the flow identified as above are forwarded between two ports of Switch 1, without going through the Router.

We also sniff RTP traces between the two computers in four scenarios to measure the quantifiable enhancement of the end-to-end RTP packet loss rate:

- NC(green) - the video call media flows through the default path, without congestion in the access network;
- NC(orange) - the video call media flows through the alternative path, without congestion in the access network;
- C(green) - the video call media flows through the default path, with congestion in the access network;
- C(orange) - the video call media flows through the alternative path, with congestion in the access network.

Table 3 summarizes one-way loss statistics of the end-to-end path: the SAVI SDI enhancement is able to decrease the packet loss rate from 1.3% to 0.0%, consistent with the media fluidity enhancement observed at user side in the screencast.

Path	**# of packets**	**# of lost**	**Loss rate**
NC(green)	408,724	0	0.0%
NC(orange)	396,600	2	0.0%
C(green)	373,055	**4959**	1.3%
C(orange)	411,762	**1**	0.0%

Table 3: Summary of one-way end-to-end loss statistics for each path for video sessions of 5 minutes long

Demo ScreenCast

We pre-recorded a screencast of a VoIP video call session 1'10" long. The guy keeps walking forth and back at constant speed to provide as a reference of the video fluidity. In the first 20", the video call is made without traffic congestion in the access network; the following 30" is the same session with congestion happening (image frozen); the last 20" is the same VoIP session with SDI enhancement operation done, while the congestion is still on going.

Conclusions

In this work, we demostrated the capability of SAVI SDI platform through a multimedia application - VoIP service. As SAVI testbed has converged with the virtual network functions to configure/reconfigure OpenFlow-enabled network elements, it provides a real-time quality enhancement solution to VoIP sessions adversely affected by congestions in the access network.

With one path reconfiguration from the platform, the end users can experience visibly improvement in media quality of their video call session. We have also seen the quantifiable improvement in end-to-end RTP packet loss rate in the point-to-point HD video call experiment.

References

[1] Shehla Abbas, Mohamed Mosbah, Akka Zemmari, and Universit Bordeaux. ITU-T Recommendation G.114, One Way Transmission Time. In *In International Conference on Dynamics in Logistics 2007 (LDIC 2007), Lect. Notes in Comp. Sciences*. Springer-Verlag, 1996.

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol, 2002.

Acknowledgements

The above works are or have been funded in part or completely by the Smart Applications on Virtual Infrastructure (SAVI) project funded under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP394424-10.

FIGURE 5.2: *Real-time Enhancement of IMS Quality of Service using SAVI SDI from Lilin Zhang, Jieyu Lin, **Hadi Bannazadeh, Alberto Leon-Garcia** at University of Toronto (Theme 5)*

5.3 An Orchestration Service of SAVI Testbed; A Heat Approach

Honbin Lu, **Hadi Bannazadeh, Alberto Leon-Garcia**, U. Toronto

Heat is an OpenStack project for providing an orchestration service to facilitate the provisioning and management of OpenStack's resources. A key feature of Heat is to provide a domain specific language (DSL) for users to declare the list of cloud resources that they need. By integrating Heat into the SAVI Testbed, users are provided with a tool to automate the resource provisioning process, and manage the complexities of application deployment and management on the Testbed.

An Orchestration Service of SAVI Testbed: A Heat Approach

Hongbin Lu, Hadi Bannazadeh, Alberto Leon-Garcia
University of Toronto, Toronto, Canada

UNIVERSITY OF TORONTO

Introduction

- Smart Applications on Virtual Infrastructure (SAVI) is a Canadian research project that explores the benefits of a hierarchical cloud.
- SAVI Testbed is an OpenStack based application platform.
- Heat provides an orchestration service for managing applications and infrastructure at SAVI Testbed

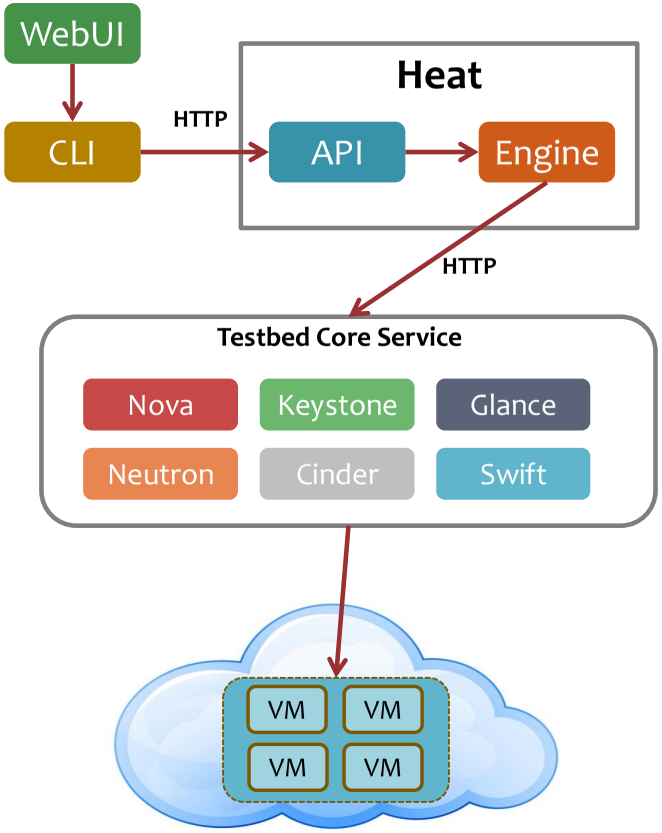
How It Works

- Users specify the cloud resources that they need to provision in a Heat template.
- Users provision cloud resources by passing the Heat template.
- Users manage the provisioned resources by using the Heat's APIs.

Resource Types

Cloud resources that are describable by a Heat template include:

- OS::Nova::Server
- OS::Nova::KeyPair
- OS::Nova::Flavor
- OS::Neutron::Net
- OS::Neutron::FloatingIP
- OS::Cinder::Volume
- OS::Neutron::SecurityGroup
- OS::Swift::Container
- …



Key Advantages

- Reduce the complexities of application deployment and management at SAVI Testbed.
- Provide a high-level Domain Specific Language (DSL) and APIs.
- Tightly Integrate with other Testbed's service.
- Provide AWS compatible APIs.

SAVI TB Workshop 2014

- Hello World
  - Provision an empty VM at SAVI Testbed.
- WordPress
  - Provision two VMs at SAVI Testbed.
  - Install a WordPress blog on the first VM.
  - Install a MySQL database server on the second VM.
  - Configure the first VM to connect to the second VM.

http://www.savinetwork.ca
Email: {hongbin,hadi, alg}@savinetwork.ca

FIGURE 5.3: *An Orchestration Service of SAVI Testbed: A Heat Approach from* **onbin Lu, Hadi Bannazadeh, Alberto Leon-Garcia** *at University of Toronto (Theme 5)*

5.4 Big Data as a Service for SAVI Testbed

Honbin Lu, **Hadi Bannazadeh, Alberto Leon-Garcia**, U. Toronto

Hadoop is a popular software framework that supports a simple programming model for Big Data processing. Sahara is a service for supporting Hadoop at OpenStack clouds. Sahara reduces the complexities of provisioning and managing Hadoop clusters by implementing the common provisioning and management routines. By integrating Sahara into the SAVI Testbed, users are provided with a tool which facilitates the provisioning of Hadoop clusters, and allows users to focus on the high-level specification of their Hadoop clusters without distracting from the low-level workflow and configurations.

Big Data as a Service for SAVI Testbed

Hongbin Lu, Hadi Bannazadeh, Alberto Leon-Garcia
University of Toronto, Toronto, Canada

UNIVERSITY OF TORONTO

Introduction

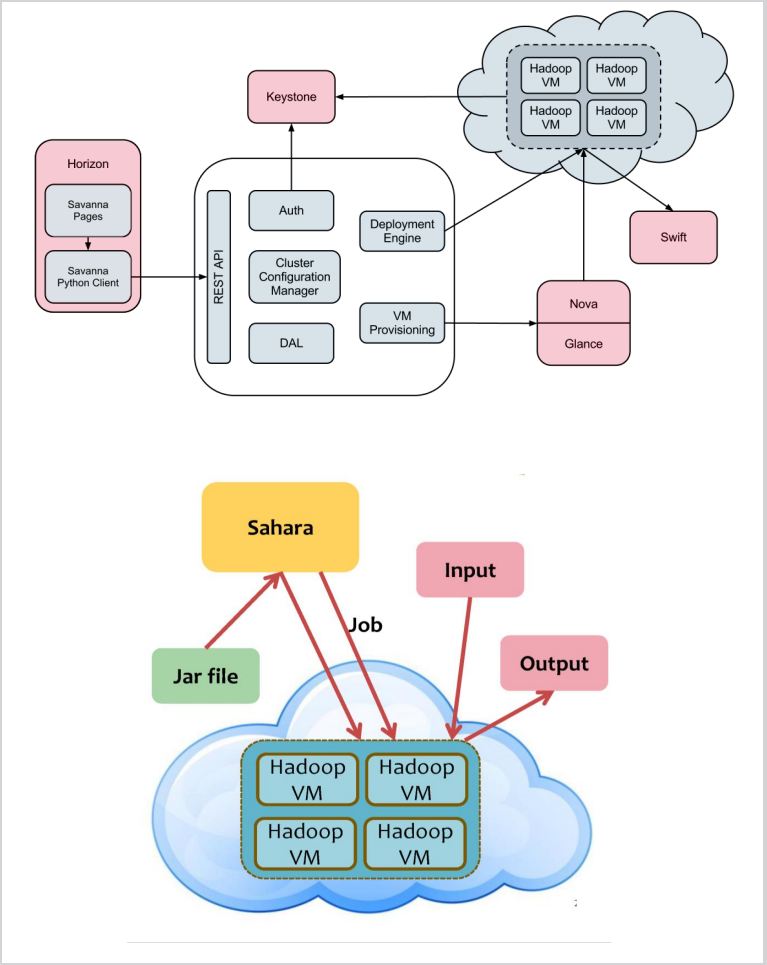
- Smart Applications on Virtual Infrastructure (SAVI) is a Canadian research project that explores the benefits of a hierarchical cloud (edge, core).
- SAVI Testbed is an OpenStack-based application platfrom.
- Sahara is a service for supporting Big Data processing at SAVI Testbed.

Features

- Simple APIs for provisioning and managing Hadoop Clusters.
- Ability to scale already provisioned clusters on demand.
- Simple APIs for submitting and monitoring Hadoop jobs.
- Support for different Hadoop distributions.

Key Advantages

- Enable users to operate on Hadoop without detailed knowledge of it.
- Fast provisioning of Hadoop clusters.
- Reduces the complexities of deploying Hadoop clusters and managing Hadoop jobs.



General Workflow

1. Provision a Hadoop cluster.
2. Create job binaries that describe the application artifacts.
3. Create data sources that describe the input and output data.
4. Define a Hadoop job.
5. Run the Hadoop job at the provisioned Hadoop cluster.
6. Get the output of the job.

Conclusion

- Sahara reduces the complexities of operating on Hadoop by implementing the common routines and providing a simple APIs.
- By integrating Sahara into SAVI Testbed, we allows users to focus on the high-level specification of their Hadoop clusters and jobs without distracting from the low-level details.

http://www.savinetwork.ca
Email: {hongbin,hadi, alg}@savinetwork.ca

FIGURE 5.4: *Big Data as a Service for SAVI Testbed from Honbin Lu,* ***Hadi Bannazadeh, Alberto Leon-Garcia*** *at University of Toronto (Theme 5)*

5.5 Cloud-RAN on SAVI; a GSM approach

Mohammad-Sina Tavoosi-Monfared, **Hadi Bannazadeh, Alberto Leon-Garcia**, U. Toronto

This poster accompanied by a demo presents an implementation of OpenBTS (Open Base Transceiver Station), a software defined GSM on SAVI Testbed. The equipment used includes an 8-channel OctoClock clock distribution module, four USRP N210 and an integrated GSPDO kit. A phone conversation with two cellular phones will be tested. To simulate C-RAN (Cloud-RAN) architecture, the transceiver module will be implemented on a closer testbed edge while the OpenBTS module (i.e. routing) will be on a different module that is not delay sensitive.

Software Based Implementation of GSM on SAVI Testbed

Sina Monfared, Hadi Bannazadeh,
Professor Alberto Leon-Garcia

Background

- ❑ A software-based GSM access point
- ❑ Allowing standard GSM-compatible mobile phones to be used as SIP endpoints in Voice over IP (VOIP) networks.
- ❑ Open source software
- ❑ Developed and maintained by Range Networks.
- ❑ The first free software implementation of the lower three layers of the industry-standard
- ❑ GSM protocol stack.
- ❑ Written in C++, works with a Remote Radio Head

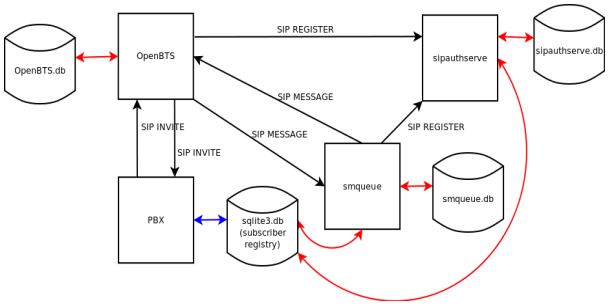
Hardware

- ➢4 x Ettus USRP-N210 (as Remote Radio Head)
- ➢GPSDO (GPS Disciplined Oscillator): For accurate UTC time (10 MHz sync clock)
- ➢Clock distributer
- ➢Datacenter (SAVI Testbed)
- ➢2 Cellular phones and SIM Cards (generic SIM card preferable)

Cloud RAN

- ❖**C-RAN**, i.e. **Cloud-RAN**, sometimes also referred as **Centralized-RAN**
- ❖A new cellular network architecture for the future mobile network infrastructure
- ❖Introduced by China Mobile Research Institute in April 2010 in Beijing, China
- ❖A centralized, cloud computing based new radio access network architecture
- ❖Large scale centralized deployment
- ❖Allows hundreds of thousands of RRH (Remote Radio Head) connect to a centralized BBU pool. The maximum distance can be 20 km in fiber link for 4G (LTE/LTE-A) system
- ❖Can support 2G, 3G systems for even larger distances

Component of OpenBTS



OpenBTS Settings Database

- ❑ OpenBTS.db is the database store for all OpenBTS configuration.
- ❑ Command Line Interface Settings:

✓GSM.Radio.Band – This is set to the GSM band appropriate for the hardware.

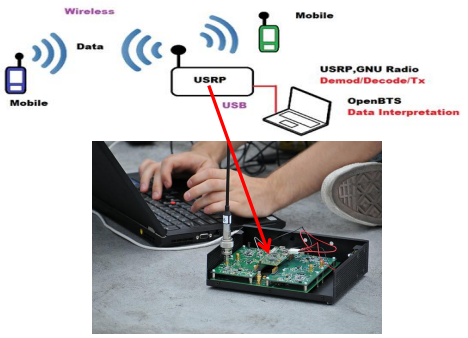
✓GSM.Radio.C0 - This is the ARFCN. This is set to something appropriate for our band.

✓Control.LUR.OpenRegistration - This is set to a regular expression matching the IMSIs of your test phones.

Other Components

- ❑ Sipauthserver: The SIP authorization server for registration traffic / Subsriber Registry. It maps the extension (i.e. cell phone number) to International Mobile Subscriber Identity (IMSI, i.e. MAC address for cell phones)
- ❑ Smqueue: The store-and-forward message service packaged with OpenBTS.
- ❑ Asterisk: It is the "standard" OpenBTS PBX (Private Branch Exchange), acting as the telephone switch. It is the most documented option, yet the most difficult (vs. FreeSwitch and Yate).

Demo



- ❑ Initially, the transceiver was run in the same VM as the all the other components
- ❑ Target:
- ➢Run transceiver on an edge closer (delay sensitive)
- ➢Run the routing software on a further edge (not delay sensitive, done in datacenter)
- ➢Inspired by C-RAN Architecture

Results

- ➢ Echo test (extension 600) works on both handsets
- ➢ The delay is not significant
- ➢ The target Cloud-RAN based architecture successfully implemented and test

Acknowledgements

❑ Special thanks to Professor Alberto Leon-Garcia
❑ Thomas Lin, Qitian Fan, Spandan Bemby, Lilin Zhang

References

[1] K.Chen et al., "C-RAN: The Road Toward Green RAN" whitepaper, China Mobile Research Institute , 2011.
[2] OpenBTS Wiki (Online) Available: https://wush.net/trac/rangepublic/wiki (Visited in June 2014)
[3] John Robb, "OpenBTS Village Base Station" in *Public Intelligent Blog*, August 2011.
[4] Gabriel Huertas, "OpenBTS Proposal" in *Wireless Sensor Network*, Octoboer 2011.

FIGURE 5.5: *Cloud-RAN on SAVI; a GSM approach from Mohammad-Sina Tavoosi-Monfared,* ***Hadi Bannazadeh, Alberto Leon-Garcia*** *at University of Toronto (Theme 5)*

5.6 Monitoring and Measurement as a Service in SDI deployed on SAVI testbed

Jie Yu (Eric) Lin, Rajsimman Ravichandiran, **Hadi Bannazadeh, Alberto Leon-Garcia**, U. Toronto

IT infrastructure monitoring and measurement is an important aspect for Intelligent management and autonomous infrastructure. In this demo, we will demonstrate the monitoring and measurement (M&M) system in the SAVI testbed. Following the Software Defined Infrastructure (SDI) concept, this system monitors both compute resources and network resources (including virtual and physical resources). It also provides a mechanism for monitoring user specified metrics (e.g. web server status). The system takes care of collecting, processing and storing all of the monitoring data including the historical records. It provides both the admin and the user visibility to the status of the infrastructure.

Monitoring and Measurement as a Service in SDI Deployed on SAVI Testbed

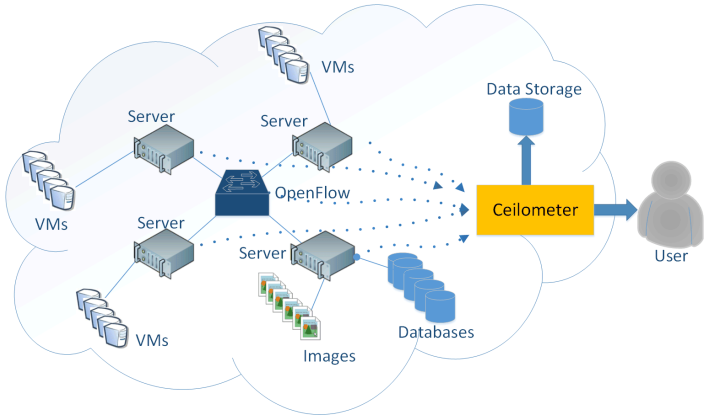
Jieyu Lin, Rajsimman Ravichandiran, Hadi Bannazadeh, and Alberto Leon-Garcia

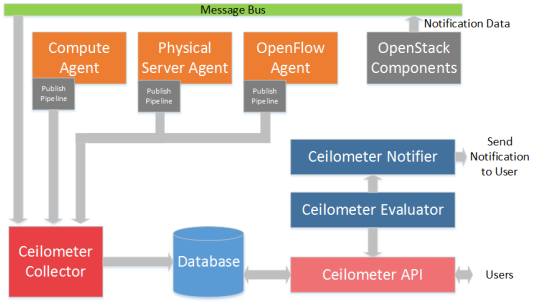
Introduction

Software Defined Infrastructure (SDI) provides a solution for converged management of heterogeneous resources in cloud infrastructures. Monitoring and measurement is an important part of SDI that facilitates converged management, as it provides support for real-time diagnostics, as well as smart applications. The monitoring and measurement system on the SAVI Testbed monitors heterogeneous resources in the cloud and provides monitoring data to users on demand. The monitoring system is built on top of Ceilometer, the telemetry component in OpenStack.

Highlights

- **Converged Monitoring:** Based on the SDI concept, the system provides converged monitoring of compute, network, and heterogeneous resources
- **Monitoring as a Service:** provides monitoring service to user on-demand
- **Scalable:** System can be configured to handle higher demand when needed
- **Expandable:** System's architecture makes it highly expandable for supporting new monitoring tasks
- **User oriented:** Capable of handling user specific monitoring tasks

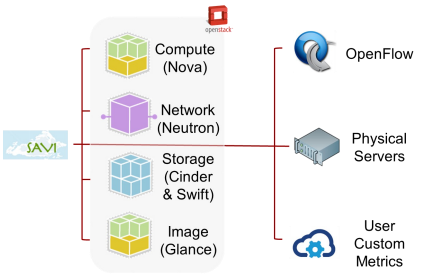




Ceilometer collects monitoring data of the virtual and physical resources in the SAVI Testbed and stores them in a Data Storage. The users can access the monitoring data using Ceilometer

Meters

Extending from Ceilometer Havana, the SAVI Testbed's monitoring system is capable of monitoring OpenStack resources (Compute, Network, Image and Storage) as well as OpenFlow for network purposes, Physical servers and user-custom metrics.



Ceilometer's Alarm feature

- A monitoring system that allows user to receive notification under certain conditions
- Triggered when one (or more) of the services crosses a threshold
- Provides multiple options for user to specify alarm trigger action. (e.g. http, send email)

Custom Monitoring

- A feature in the system that allows user to monitor custom metrics (e.g. requests/sec in web server, processes running in VM)
- To utilize Custom Monitoring feature, a User Agent can be installed in a VM and configured to monitor the metrics interested
- User Agent retrieves monitoring data and delivers them to the Testbed's monitoring infrastructure for processing and storage

Future Work

- Offer data analytics capability to facilitate real-time diagnostics and converged management
- Stream processing capability
- Improve fault tolerance
- User interface for visualization

FIGURE 5.6: *Monitoring and Measurement as a Service in SDI deployed on SAVI testbed from Jie Yu (Eric) Lin, Rajsimman Ravichandiran,* ***Hadi Bannazadeh, Alberto Leon-Garcia*** *at University of Toronto (Theme 5)*

5.7 Efficient Multicast algorithm on SAVI network

Sai Qian Zhang, **Hadi Bannazadeh, Alberto Leon-Garcia**, U. Toronto

Group communication is widely applied in the daily computer network system. When more than one receiver is considered in the data transmission mechanism, multicast is the most efficient and reliable solution. In this poster, we propose an efficient multicast algorithm used among different virtual machines (VMs) in the SAVI network. The algorithm best fits for the dynamic network topology so that when a new VM joins the multicast network and the topology changes, we do not need to reconfigure the whole multicast tree topology. In particular, the multicast tree is constructed based on the following objectives: 1) Saving Bandwidth & 2) Minimizing Delay

Efficient multicast on SAVI network

Sai Qian Zhang, Hadi Bannazadeh, Alberto Leon-Garcia
Department of Electrical and Computer Engineering, University of Toronto

Contact Information:
Electrical and Computer Engineering
University of Toronto
Phone: +1 (647) 938 1366
Email: sai.zhang@mail.utoronto.ca

Abstract

Group communication is widely used in computer network systems. When more than one receiver is involved in the data transmission mechanism, multicast is the most efficient and reliable solution. In this project, we propose an efficient multicast algorithm with the goal of saving bandwidth and minimizing transmission delay on SAVI testbed. We have already implemented this multicast mechanism on the SAVI edge network using the API provided by SDI manager.

Introduction

Group communication is widely used in modern communication networks. Multicast lends itself naturally to these communication patterns. In multicast communication, messages are concurrently sent to multiple destinations, all members of the same multicast group. Such communication mechanisms are becoming increasingly important for datacenter networks such as Hadoop, GFS, clustered application servers. The former multicast techniques on SAVI network are not efficient and robust. The implementation of this multicast mechanism will help to save link bandwidth, decrease transmission delay and avoid network congestion.

Different kinds of multicast algorithms have been proposed and evaluated. Since the problem of finding the steiner tree is an NP-hard problem, heuristic algorithms have been proposed. [1] provides an algorithm to finding a multicast tree with a constraint on end-to-end delay but the bandwidth utilization is not promised. [2] shows a heuristic algorithm to generate the shortest path tree with the lower path concentration. [3] proposed an SDN-based system that enables multicast in commodity switches used in the data centers. However, both [2] and [3] do not consider the end-to-end delay of the multicast transmission. To mitigate the effect of wasting bandwidth and transmission delay, we propose a new multicast algorithm which builds a shortest path tree with end-to-end delay constraint.

In this project, the following contributions have been made:

1. An efficient multicast algorithm that aims to jointly save bandwidth and decrease transmission delay has been developed.
2. By using the API provided by SDI manager, we can get the topology of the edge network and the detailed information of each VM, which is used as the input to the multicast mechanism.
3. The multicast mechanism implements the flow on the edge network by using the API provided by SDI manager.

Project Goal and Objectives

1. To develop an efficient multicast mechanism on the SAVI edge network
2. Find the most suitable multicast tree with the purpose of saving bandwidth and minimizing transmission delay.

SAVI smart edge topology

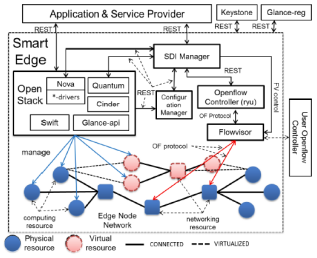


Figure 1: SAVI smart edge topology

SAVI multicast topology on smart edge

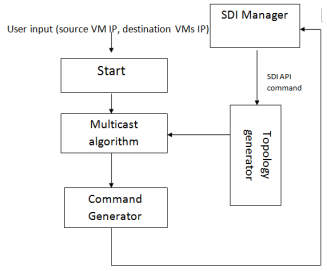


Figure 2: SAVI multicast mechanism on smart edge

- **Start** provides the user interface, user can specify the ip address of source VM and multicast destination VMs in this file.
- **Topology generator** generates the edge network topology by using the API provided by SDI manager.
- **Multicast algorithm** takes the user input and the network topology, runs the routing algorithm and sends the resulting multicast tree to **Command Generator**
- **Command Generator** takes the input from the **Multicast algorithm** and outputs the respective API commands.

Multicast routing algorithm

The multicast routing algorithm can be best described by using the following pseudo commands:

Algorithm: Multicast Routing Algorithm

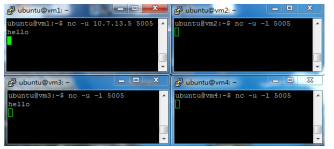
Procedure Multicast(edge network, src node, dst nodes,threshold)
  multicast-tree = [src node]
  m=size(dst nodes)
  for $(i = 1; i <= m; i++)$
    path list = $findthetopkshortestpath$(edge netwok,multicast-tree, dst nodes[i])
    $sort$(path list)
    lower limit = positive infinity
    for(each path **in** path list)
      if($totaltranmissiondelay$(path)$< threshold$)
        multicast-tree.append(path)
        break
      end **if**
      delay=$totaltranmissiondelay$(path)
      if $(delay < lowerlimit)$
        lower limit=delay
        candidate path=path
      end **if**
    end **for**
    multicast-tree.append(candidate path)
  end **for**
end **function**

Table 1: Multicast routing algorithm

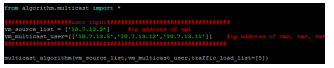
$findthetopkshortestpath$(edge network, multicast-tree, dst node[i]) returns the top k shortest path from any node in the existing multicast tree to the dst node [i] , this can be implemented by using the Yen's algorithm. $totaltransmissiondelay$(path) returns the transmission delay of the path. $sort(pathlist)$ sorts the path in the path list by their length.

Demo

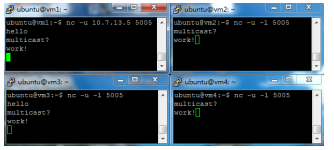
This section gives the screen shots of the demo. To demo the multicast mechanism, we deploy four virtual machines on the edge network named VM1, VM2,VM3 and VM4. VM1 is used as source VM, and the rest VMs are destination VMs. We use netcat command to send udp packet to VM3. Before the multicast mechanism is enabled, only VM3 can receive the udp packet.



Here is the user interface of the multicast mechanism, all the VMs are represented by their ip address.



After the execution, the packet sent by VM1 can reach all the destination VMs.



Conclusions

- We propose an efficient multicast mechanism and implement on SAVI network by using the API provided by SDI manager.
- We build the multicast tree with the purpose of saving bandwidth and minimizing transmission delay.

Reference

[1] G.Rouskas,I.Baldine, S. Radia. "Multicast routing with end-to-end delay and delay variation constraints". INFOCOM,1996

[2] L. Wei and D.Wstrin. "A comparison of Multicast Trees and Algorithms". INFOCOM, 1994

[3] A.Iyer, P.Kumar,V.Mann. "Avalanche:Data Center Multicast using Software Defined Networking". COMSNETS,2014

FIGURE 5.7: *Efficient Multicast algorithm on SAVI network from Sai Qian Zhang,* ***Hadi Bannazadeh, Alberto Leon-Garcia*** *at University of Toronto (Theme 5)*

5.8 End-to-End Traffic Control in the SAVI Testbed

Thomas Lin, **Hadi Bannazadeh, Alberto Leon-Garcia**, U. Toronto

This demo will show an example of how the SAVI testbed enables end-to-end traffic control as managed by the user. For the purposes of this demo, we have integrated a Wi-Fi access point (WAP) that also allows mobile devices to be integrated into a virtual network. The SAVI testbed currently enables users to control their own virtual networks using an OpenFlow controller running with a publicly accessible IP address. The demo will showcase a scenario in which the mobile device is connected to the testbed and receives a live video stream from a virtual server running within the testbed, while the network is being congested. With the user's own OpenFlow controller the user is able to direct traffic through a pre-configured queue that guarantees a certain level of available bandwidth. The demonstration will aim to show the effect of queuing on the quality of the displayed video on the mobile device.

End-to-End Traffic Control in the SAVI Testbed

Thomas Lin, Hadi Bannazadeh, Alberto Leon-Garcia
University of Toronto

Introduction

The SAVI testbed aims to present an experimental environment upon which research can be conducted on Future Internet protocols and applications. To facilitate this goal, the SAVI testbed provides the ability for users to control the network forwarding logic of their own slice. This work presents a first step towards extending that control to include bandwidth reservation and traffic prioritization.

Motivation

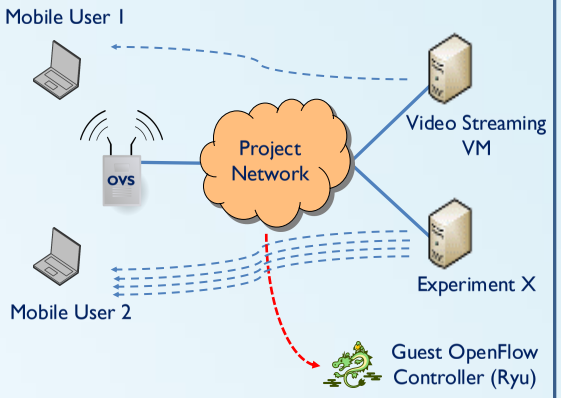
- Many users are concurrently running experiments on the testbed
- 1 & 10 GE testbed network fabric
- 10 GE-capable hardware resources (e.g. NetFPGAs, DE5-Nets, etc.) in the testbed
- Quality of Service features needed to satisfy bandwidth and delay-sensitive applications
- **Long term goal:** Enable network forwarding control applications based on network bandwidth availability

Background

- Whale (SAVI topology manager) enables users to query topology, switch, link, and port information & statistics
- Open vSwitch supports basic traffic policing, classification, and queueing using Linux HTB

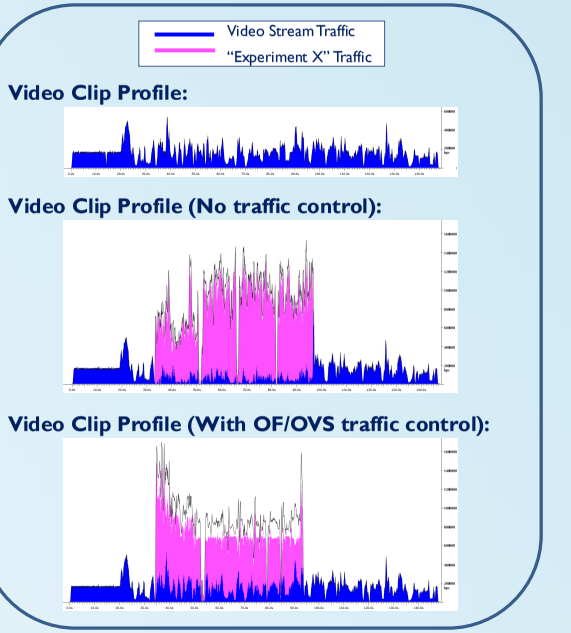
Simulation Setup

- Wi-Fi access points running OpenWrt firmware and Open vSwitch within
- Access points are connected to the testbed enabling mobile devices to join project networks



- Forwarding logic of project network (a "slice") delegated to a guest OpenFlow controller via FlowVisor*
- Mobile User 1 streams video from VM within the testbed
- Mobile User 2 participates in high-bandwidth experiment with secondary VM within the testbed

Simulation Results



Future Work

- Integration of link bandwidth information into Whale from SAVI's monitoring and measurement system
- Enabling bandwidth-aware network control applications

* Custom-compiled FlowVisor to fix bug regarding OpenFlow's Enqueue action: https://github.com/t-lin/flowvisor

Acknowledgements to Heming Wen, Kevin Han, and Michael Smith for the previous work on Wi-Fi AP integration into SAVI

FIGURE 5.8: *End-to-End Traffic Control in the SAVI Testbed from Thomas Lin,* ***Hadi Bannazadeh, Alberto Leon-Garcia*** *at University of Toronto (Theme 5)*

5.9 L2/L3 Overlay Software Defined Network on SAVI Testbed

Khashayar Hoseinzadeh, **Hadi Bannazadeh, Alberto Leon-Garcia**, U. Toronto

In this work, we will utilize the SAVI testbed alongside a software defined switch (OpenVSwitch) to dynamically create network topologies specified by the user. Similar to mininet, the users will be able to specify the number of hosts and switches and specify how they want them to be connected. The goal is to create a layer 2 or a layer 3 network, linking up the nodes as defined by the user topology. For this purpose, we use the SAVI Test-Bed and VXLAN tunneling protocol to create arbitrary overlay networks. These overlay networks could be used in variety of applications from deploying a large scale OpenFlow network in order to test a SDN controller (and application) to running a large scale distributed application on a desired topology.

Maxinet: L2/L3 Overlay Using Software Defined Networking

Khashayar Zadeh, Hadi Bannazadeh, Spandan Bemby, Alberto Leon-Garcia
University of Toronto, Toronto, Canada

UNIVERSITY OF TORONTO

SAVI

Introduction

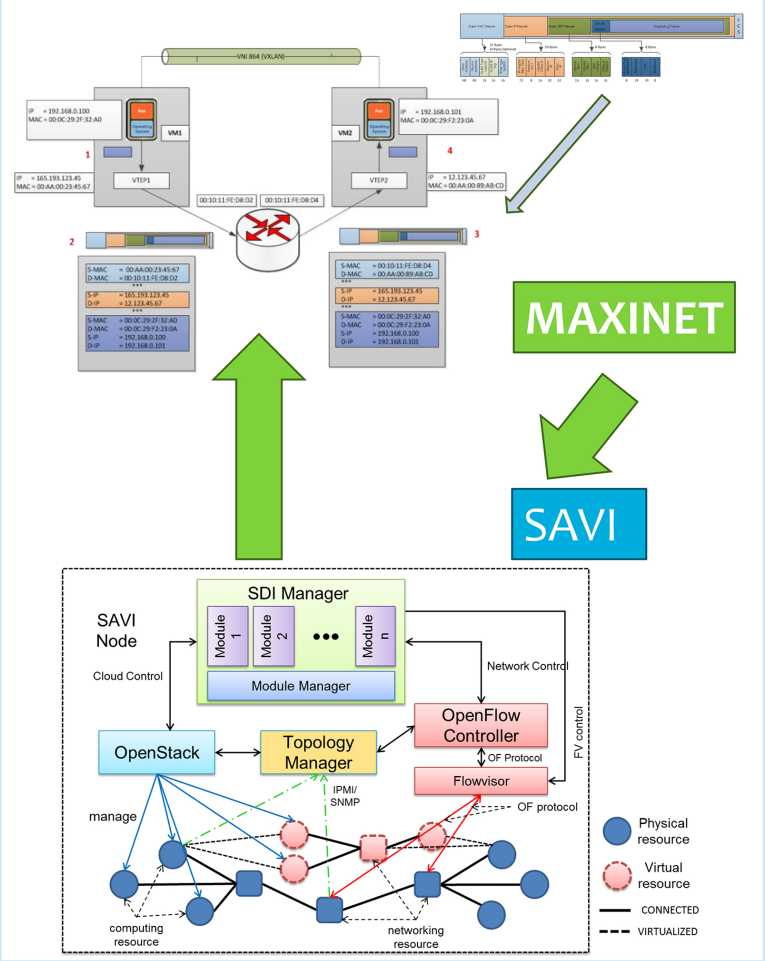
- Create a private OpenFlow network on top of the SAVI testbed.
- Opportunities to test Software Defined Network controllers and applications.
- The ability to test the distribution of large scale applications on any arbitrary topology.

Key Advantages

- Creates a network, with realistic throughput and delays, with real kernels, switches and applications.
- Uses VXLAN to create logical networks.
- Compared to VLAN, VXLan allows 16 million network IDs - which can be used to create very large and complex topologies.

Use Case

- By specifying topology in a framework agnostic way, we can parallel prototype topology on SAVI and Amazon EC2.
- Create more reliable distributed systems: change backend server by changing IP and MAC Address of host via OVS.
- Allow for live migration between incompatible cloud infrastructures, like SAVI and EC2.



Architecture

- User specifies authentication credentials and topology.
- Maxinet creates VMs, Open vSwitches; connects components as desired .
- Uses VXLAN to create a logical network for VMs across different networks.
- VXLAN tunneling encapsulates L2 Ethernet frames in L3 UDP Packets, to connect OVS switches directly at L2.

Future Work

- Separate topology specification from booting VMs and connecting network, to use with Amazon EC2.
- Use YAML to specify topology.
- Integrate with Openstack Heat.

http://www.savinetwork.ca
Email: {khash,hadi,spandan,alg}@savinetwork.ca

FIGURE 5.9: *L2/L3 Overlay Software Defined Network on SAVI Testbed from Khashayar Hoseinzadeh,* ***Hadi Bannazadeh, Alberto Leon-Garcia*** *at University of Toronto (Theme 5)*

Page for Notes