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.
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.

## 1 Theme 1 Smart Applications

1.1 Soda: A Case for Software-Defined Networking at the Application Layer ........................................ 2
1.2 Consistency Models in Distributed Controller Architectures for Software Defined Networks ............ 4
1.3 Repairing Erasure Codes Cooperatively in Storage-Intensive Applications ................................. 6
1.4 Ready, Set, Go: Coalesced Offloading from Smart Applications to the Cloud ............................... 8
1.5 Blossom: Content Distribution across Multiple Geographic Locations ......................................... 10
1.6 Towards an Architecture for Mobile Social Video-Sharing Apps .................................................. 12
1.7 A Case Study on Multimedia Gaming Applications in Clouds ....................................................... 14
1.8 Smart Applications with Green Deployments ..................................................................................... 16
1.9 Slicing Applications in the SAVI Pie .................................................................................................. 18
1.10 MobileView: Optimized Mobile Video Conferencing over the SAVI Testbed ............................... 20
1.11 GLINT – A Horizon based Image Distribution System ................................................................. 22

## 2 Theme 2 Extended Cloud Computing

2.1 Task Assignment Across Clouds by Graph Partitioning ................................................................. 26
2.2 Simulation of Future Application-Aware Multi-Clouds .................................................................... 28
2.3 Relational Edge Caching for Edge-Aware Web Applications ......................................................... 30
2.4 Optimal Service Replica Placement via Predictive Model Control ............................................... 32
2.5 An Architecture for Mitigation Low and Slow Application DDoS Attacks .................................... 34
2.6 Model-driven Elasticity and DoS Attack Mitigation in Cloud Environments .................................... 36
2.7 Cloud bursting for MapReduce jobs: A dream or a reality? ............................................................ 38
2.8 Crowd-sourcing Sensor Data ............................................................................................................ 40

## 3 Theme 3 Smart Converged Edge

3.1 On Satisfying Green SLAs in Distributed Clouds ............................................................................ 44
3.2 CQNCR: Optimal VM Migration Planning in Cloud Data Centers .............................................. 46
3.3 Design and Management of DOT: A Distributed OpenFlow Testbed .............................................. 48
3.4 A Simple Programming Model for Scalable SDN Control Applications ......................................... 50
3.5 FleXam: Flexible Sampling Extension for Monitoring and Security Applications in OpenFlow .... 52
3.6 Dynamic Virtual Infrastructure Provisioning in Geographically Distributed Clouds ..................... 54
3.7 Online Algorithms for Energy Cost Minimization in Cellular Networks ........................................ 56
3.8 Greenhead: Virtual Data Center Embedding Across Distributed Infrastructures .......................... 58

## 4 Theme 4 Integrated Wireless/Optical Access

4.1 Improving Throughput and Fairness in Virtualized 802.11 Networks through Association Control .................................................. 62
4.2 QoS-Oriented Slice Provisioning in Wireless Virtualized Networks ............................................. 64
4.3 Management of Virtualized and Software-Defined Wireless Infrastructures: Issues, Requirements, Framework and Specifications .................................................. 66
4.4 Aurora: A Virtualization and Software-Defined Infrastructure Framework for Wireless Networks .... 68
4.5 Full-duplex WiFi Analog Signal Transmission with Digital Signal Downlink in Radio-over-Fiber System Employing RSOA-based WDM-PON Architecture .................................. 70
4.6 An Application of Aurora with 802.11 wireless networks and Flow-based Virtualization 72

5 Theme 5 SAVI Application Platform Testbed 75

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Virtualized Reconfigurable Hardware in the SAVI Testbed</td>
<td>76</td>
</tr>
<tr>
<td>5.2 Real-time Enhancement of IMS Quality of Service using SAVI SDI</td>
<td>78</td>
</tr>
<tr>
<td>5.3 An Orchestration Service of SAVI Testbed; A Heat Approach</td>
<td>80</td>
</tr>
<tr>
<td>5.4 Big Data as a Service for SAVI Testbed</td>
<td>82</td>
</tr>
<tr>
<td>5.5 Cloud-RAN on SAVI; a GSM approach</td>
<td>84</td>
</tr>
<tr>
<td>5.6 Monitoring and Measurement as a Service in SDI deployed on SAVI testbed</td>
<td>86</td>
</tr>
<tr>
<td>5.7 Efficient Multicast algorithm on SAVI network</td>
<td>88</td>
</tr>
<tr>
<td>5.8 End-to-End Traffic Control in the SAVI Testbed</td>
<td>90</td>
</tr>
<tr>
<td>5.9 L2/L3 Overlay Software Defined Network on SAVI Testbed</td>
<td>92</td>
</tr>
</tbody>
</table>
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.
**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.

**Utility Max-Min Fair Allocation**

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

$$\max \min\limits_{i \in V} \phi(\sum\limits_{j \in P_i} b_{ij})$$

s.t.

$$\sum_{j \in P_i} b_{ij} = \phi_i^{-1}(d_{ij}), \quad \forall k \in (1, \ldots, K - 1), \quad i \in B^k, \quad b_{ij} \geq 0, \quad \forall j \in P_i, \forall i \in V, \quad \sum_{j \in P_i} b_{ij} \leq D_i, \quad \forall i \in V, \quad \phi_i(\sum_{j \in P_i} b_{ij}) \text{ are equal}, \quad \forall i \in V, \quad \sum_{i \in F} \phi_i(b_{ij}) \leq C_i, \quad \forall i \in L.$$

**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.

**Integrate Pathlet Routing into Soda**

**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

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 physical 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.

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.

\[
\begin{array}{ccc}
A & A & A \\
B & B & B \\
A+B & A+2B & \\
\end{array}
\]

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 \(\text{eg. over switches}\) 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 number of existing blocks needed to repair the original data
  - \(d\) the number of blocks that can recover the original data
  - \(t\) the number of failures repaired at the same time

- CR codes are constructed in two parts:
  - Systematic minimum-storage regenerating (MSR) codes
    - \(d - k + 1\) symbols
    - \(k(d - k + 1)\) symbols
  - \((n, k - 1)\) MDS codes
    - \(\sum_{i=1}^{k} h_i C_i\)
    - \(i = 1, 2, ..., n\)

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

- CR codes can be repaired in a cooperative way:

\[
\begin{align*}
F & \leftarrow^\uparrow_{\text{repair bandwidth in total (GB)}} \left( F + \sum_{i=1}^{k-1} h_{ij} C_i \right) \\
\text{rank} & = k - 1 \\
\end{align*}
\]

\[
\begin{align*}
\sum_{i=1}^{k-1} h_{ij} (v_{i,j}^F + w_{i,j} v_{i,j}^F) & \leftarrow h_{ij} F \\
\text{rank} & = d - k + 1 \\
\end{align*}
\]

Performance Evaluation

- Repair bandwidth in total (GB)
- Disk I/O

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 NETGF394424-10.

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 “bundled," 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 $g_1, g_2, \ldots, g_9$. 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:

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

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 $A_0$ is defined as a randomized algorithm that selects $\theta$ 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 $g_1, g_2, \ldots, g_\theta$ be the times at which requests are granted and transmitted. Algorithm $A_0$ grants the next request at $g_{\theta+1}$ such that there exists a time $\tau_{\theta+1}, g_\theta < \tau_{\theta+1} < g_{\theta+1}$, that satisfies

$$R(g_i, \tau_{i+1})(\tau_{i+1} - g_i) = \left(\frac{\theta}{\alpha}\right)S_i$$

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 $\tau_{i+1}$ 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 $A_0$ 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 $\alpha$. 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 paper, 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-ε)-optimality. In particular, we develop an algorithm that is 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.

\[ S = \{(D_1, D_2), D_1, D_2, D_1, D_2, D_1, D_2\} \]

- 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.

\[ \text{P2: } \max_{\mathbf{x}, \mathbf{y} \in F} \langle \mathbf{y} \rangle \]

subject to

\[ f(x) \geq \lambda_i \cdot \text{dem}_i, \forall i \]

\[ \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \cdot \delta(t_j) \leq c_i, \forall e \in E \]

\[ \lambda_i \geq 0, x_{ij} \geq 0, \forall i, \forall j \]

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.

\[ \text{Theorem 1. The running time of Blossom is}\]

\[ O((c^{1+\epsilon})/(2\alpha n^{1+\epsilon} + n\epsilon^2||E||)) \]

\[ \text{Theorem 2. The resulted satisfaction ratio vector } \langle \lambda^* \rangle \text{ is indeed max-min fair.} \]

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]

<table>
<thead>
<tr>
<th>VA</th>
<th>CA</th>
<th>EU</th>
<th>SG</th>
<th>BR</th>
<th>OR</th>
<th>JP</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>28</td>
<td>26</td>
<td>19</td>
<td>19</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>31</td>
<td>24</td>
<td>31</td>
<td>19</td>
<td>65</td>
<td>43</td>
<td>71</td>
</tr>
<tr>
<td>110</td>
<td>9.65</td>
<td>21.7</td>
<td>13.8</td>
<td>46.8</td>
<td>7.15</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>27.9</td>
<td>16</td>
<td>9.42</td>
<td>15</td>
<td>20.8</td>
<td></td>
</tr>
<tr>
<td>25.4</td>
<td>85.4</td>
<td>25.9</td>
<td>11.8</td>
<td>51.4</td>
<td>31.4</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>35.9</td>
<td>16.9</td>
<td>12.4</td>
<td>58.8</td>
<td>54.4</td>
<td></td>
</tr>
</tbody>
</table>

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 RDBMS 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.
BACKGROUND

The typical 3-tier, SOA, web-based application architecture is evolving: (a) they have multiple mobile clients collecting multimedia data and spatial, temporal and social context. (b) they require adaptivity through analytics on cloud-based NoSQL repositories.

In this poster, we describe an architecture for developing such types of systems, on the SAVI cloud.

REAL-TIME SERVICES

- Peer-group discovery:
  - Who is close to here now?
- Social-Network-based Multimedia Content Sharing:
  - Peer-to-peer (P2P) overlay streaming.

STORAGE

NoSQL cloud-based repositories:
- Different types for different content:
  - HBase for almost-relational data
  - MongoDB for documents
  - Neo4j for graphs

CLOUD-BASED ANALYTICS

- Massively parallel processing for feature extraction from raw content:
  - OpenCV (with SIFT and MSER) to automatically extract image and video features.
- Scalable pattern mining and machine learning:
  - Mahout
  - BigInsights

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. Cybera has provided infrastructure support.
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

- 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.

Game Flow

Future Work
- A PaaS that will enable developers to compose multi-party games using third-party substrates.

Acknowledgment
SAVI (NETGP394424-10)
FQRNT - nouveau chercheur

References
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

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

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?

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.
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

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.

Oregon Users (West Coast)

Since Carleton is further away from 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.

Virginia Users (East Coast)

Since Victoria is further away from 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 1: Connection Setup for Potentially Low Latency Scenario.

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

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.
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

Mean=95.6 ms

95th percentile =146.8 ms
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.
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.

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.
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 to the task graph to the edge or core by minimizing a cut related to communication volumes and processing capacity.

Extended cloud architecture

Objectives

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

Constraints

- Response time limit
- Host processor capacity
- Memory limit

Why is low-latency needed?

- Characteristics of future applications - 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

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 G1, G2, ..., Gm
- 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
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

- Begin
- Find tasks to be assigned
- End

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

Systematic Functional Models

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 NETP034450-10

Figure 2.1: Task Assignment Across Clouds by Graph Partitioning from Ravneet Kaur, Dr. John W. Chinneck, Dr. C. Murray Woodside.
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 to 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.

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.

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

Results
After simulating 24 hours

Comparison Of Simulators

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

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

Acknowledgements

Figure 2.2: Simulation of Future Application-Aware Multi-Clouds from Derek Hawker, John Chinneck, Murray Woodside.
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

**Introduction**

- With the global expansion of web applications like eBay, Craigslist and Groupon, data centers have become more distant from the users.
- 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.

**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 labels 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 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.
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 Ghanbari1, Marin Litoiu1, Przemyslaw Pawluk1, Cornel Barna1

1Dept. of Computer Science, York University, Toronto

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 of some service replicas to be added or removed from the available hosts. These deployment changes are optimal with regard to overall objectives defined over time.

Proposed Method

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

\[ \text{given: } R_{j}^{N}(x) \text{ for each } c, j, \theta_{s,j} \text{ for each } s, h; \Omega_{h} \text{ for each } h, \]

\[ \text{minimize } \sum_{s} \sum_{h} \sum_{j} \left( \sum_{i} \theta_{s,j} \left( X_{j}^{N}(x_{i}) - N_{j}^{c} \right) \right) \]

\[ + \sum_{s} \sum_{h} \sum_{j} \theta_{s,j} \sum_{i} \sigma_{c} \left( x_{i} - d_{i} - \theta_{s,j} \right) \theta_{s,j} \]

\[ + \text{subject to: } \sum_{i} \theta_{s,j} = 1 \text{ for each } s, j \]

\[ \theta_{s,j} \geq 0 \text{ for each } s, j \]

\[ \sum_{h} \theta_{s,j} \leq \Omega_{h} \theta_{s,j} \text{ for each } s, j \]

\[ U_{s,j} = \sum_{i} \sum_{j} d_{i} \theta_{s,j} \text{ for each } h, j \]

\[ U_{s,j} \leq \Omega_{h} \theta_{s,j} \text{ for each } h, j \]

Nomenclature

\[ \Omega_{h} \] The multiplicity of host \( h \)
\[ r_{\text{opt}} \] The trashing cost coefficient
\[ r_{\text{cost}} \] The resource cost coefficient
\[ r_{\text{SLA}} \] The SLA violation cost coefficient
\[ \sigma_{c}^{h} \] A host-service specific cost coefficient based on utilization
\[ \sigma_{h} \] The speed factor of host \( h \)

\[ \theta_{s,j} \] The portion of service \( s \) that is handled by host \( h \) at time \( t \) through a placed replica and proper routing.

U_{s,j} \text{ Utilization of a resource } j \text{ of a host } h

C The number of classes of customers in the workload
\( d_{i,j} \) 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)
\( N_{j}^{c} \) Predicted number of users in class \( c \) at time \( t \)
\( pos \left( a \right) \) Positive value of a real number or \( \max \left( a, 0 \right) \)
\( SLA_{j} \) SLO upper bound on the response time of class \( c \) at time \( t \).
\( S \) The number of services in the datacenter
\( t \) Current time step index
\( n_{s,j} \) Change in the placement of the service replica of the service \( s \) on host \( h \) made at time \( t \).

X_{j} \text{ Average system throughput for class } j \text{ at time } t

\( X_{j}^{N} \) SLO upper bound on the throughput of class \( c \) at time \( t \)
\( Z_{j} \) SLA of class \( c \) at time \( t \)

Experiment Set 1

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

Experiment Set 2

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 of some service replicas to be added or removed from the available hosts. These deployment changes are optimal with regard to overall objectives defined over time.

Proposed Method

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

\[ \text{given: } R_{j}^{N}(x) \text{ for each } c, j, \theta_{s,j} \text{ for each } s, h; \Omega_{h} \text{ for each } h, \]

\[ \text{minimize } \sum_{s} \sum_{h} \sum_{j} \left( \sum_{i} \theta_{s,j} \left( X_{j}^{N}(x_{i}) - N_{j}^{c} \right) \right) \]

\[ + \sum_{s} \sum_{h} \sum_{j} \theta_{s,j} \sum_{i} \sigma_{c} \left( x_{i} - d_{i} - \theta_{s,j} \right) \theta_{s,j} \]

\[ + \text{subject to: } \sum_{i} \theta_{s,j} = 1 \text{ for each } s, j \]

\[ \theta_{s,j} \geq 0 \text{ for each } s, j \]

\[ \sum_{h} \theta_{s,j} \leq \Omega_{h} \theta_{s,j} \text{ for each } s, j \]

\[ U_{s,j} = \sum_{i} \sum_{j} d_{i} \theta_{s,j} \text{ for each } h, j \]

\[ U_{s,j} \leq \Omega_{h} \theta_{s,j} \text{ for each } h, j \]

Nomenclature

\[ \Omega_{h} \] The multiplicity of host \( h \)
\[ r_{\text{opt}} \] The trashing cost coefficient
\[ r_{\text{cost}} \] The resource cost coefficient
\[ r_{\text{SLA}} \] The SLA violation cost coefficient
\[ \sigma_{c}^{h} \] A host-service specific cost coefficient based on utilization
\[ \sigma_{h} \] The speed factor of host \( h \)

\[ \theta_{s,j} \] The portion of service \( s \) that is handled by host \( h \) at time \( t \) through a placed replica and proper routing.

U_{s,j} \text{ Utilization of a resource } j \text{ of a host } h

C The number of classes of customers in the workload
\( d_{i,j} \) 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)
\( N_{j}^{c} \) Predicted number of users in class \( c \) at time \( t \)
\( pos \left( a \right) \) Positive value of a real number or \( \max \left( a, 0 \right) \)
\( SLA_{j} \) SLO upper bound on the response time of class \( c \) at time \( t \).
\( S \) The number of services in the datacenter
\( t \) Current time step index
\( n_{s,j} \) Change in the placement of the service replica of the service \( s \) on host \( h \) made at time \( t \).

X_{j} \text{ Average system throughput for class } j \text{ at time } t

\( X_{j}^{N} \) SLO upper bound on the throughput of class \( c \) at time \( t \)
\( Z_{j} \) SLA of class \( c \) at time \( t \)

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.
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

Performance Model-Based Approach

- LSDoS Detection Sensor
  - Two phases:
    1. Build a Baseline
    2. Compare Metrics

- 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

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

COTS Components

- LSDoS 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

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

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
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

---

**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 automatically 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**

A 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.

**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 Estimator component to estimate new values for the parameters.

---

**METHOD**

1. Extract metrics
2. Did a service violation occur?
   - yes
   - no
3. Extract filtered scenarios
4. Did an SLA violation occur?
   - yes
   - no
5. Calculate n (servers to add)
6. Try to add the scenario
   - no
   - yes
   - Add n servers
7. Perform Scenarios

---

**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.

---

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 MapReduce (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
<table>
<thead>
<tr>
<th>MR Burst Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
</tr>
<tr>
<td>Cost Model</td>
</tr>
<tr>
<td>Performance Model</td>
</tr>
</tbody>
</table>

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.

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)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>T (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>grep</td>
<td>1.06</td>
</tr>
<tr>
<td>term vector</td>
<td>4.77</td>
</tr>
<tr>
<td>inverted index</td>
<td>6.21</td>
</tr>
</tbody>
</table>

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.
Introduction:
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 practical steps, we build a mobile application that interacts with a data management system to advise travel conditions. Our aim is to alleviate transportation issues faced by the commuters and the government by analyzing the data obtained by crowd-sourcing of mobile users.

Crowd-sourcing Sensor Data
Saeed Zareian, Manish Garg, Mark Shtern, Rizwan Mian, Marin Litoiu
York University, Toronto, Canada

Application:
- Road Traffic Status
- Finding Bumps on the Road
- Showing Slow Traffic
- Prediction of Next Destination
- Keeping User History of Routes
- Showing Stop Points and Destinations

Time Series Storage in HBase:
- More efficiency in storage and retrieval.
- Special column and key design

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 threshold. 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\textsuperscript{1,2}, Mohamed Faten Zhani\textsuperscript{2}, Qi Zhang\textsuperscript{2}, Rami Langar\textsuperscript{1} and Raouf Boutaba\textsuperscript{2}

\textsuperscript{1}Pierre and Marie Curie University, Paris VI, France, \textsuperscript{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 a 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

**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

**Problem Formulation**

- **System Model**
  - Decision variables: VM and virtual link mapping over time
  - \[ x_{jk}^t = \begin{cases} 1 & \text{if the VM } k \text{ of the VDC } j \text{ is assigned to data center } i \text{ during slot } t \\ 0 & \text{otherwise} \end{cases} \]
  - \[ f_{ik}^t = \begin{cases} 1 & \text{if link } k \in E \text{ is used to embed virtual link } e^t \text{ during slot } t \\ 0 & \text{otherwise} \end{cases} \]

- **Objective Function**
  - \[ \text{Maximize } R_k - (D_k + B_k + M_k + P_k) \]

  \[ k \text{-reporting period} \]
  \[ R \text{- Revenue, } D \text{- cost in data centers, } B \text{- inter-data center bandwidth cost, } M \text{- migration cost, } P \text{- penalty (SLA violation)} \]

**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
  - Impact of the reconfiguration interval
  - 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
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%.
CQNCR: Optimal VM Migration Planning in Cloud Data Centers

Md. Faizul Bari, Mohamed Faten Zhani, Qi Zhang, Reaz Ahmed, and Raouf Boutaba
David R. Cheriton School of Computer Science, University of Waterloo

VM Migration: Why do we need it?
- Virtual Machine (VM) migration enables dynamic resource reconfiguration
- Reducing operational costs
- Maximizing resource utilization
- Improving performance & reliability

A large number of VM migrations can cause:
- Service disruption
- High resource consumption
- Network congestion
- Higher provisioning time

VM Migration Sequencing Problem
- Given an initial and a target virtual-to-physical mapping, the goal is to find an optimal migration sequence that
- Reduces total migration time
- Minimizes service disruption time
- Avoids server overload
- Reduces network congestion

Example: initial & target embedding

Problem Formulation
- We formulate VM Migration Sequencing Problem as an ILP
- Objective function

Example: impact of migration sequence

Proposed Heuristic: CQNCR
- Finding the migration sequence
  - Step 1: Find VMs that can be migrated at the current time
  - Step 2: Group these VMs into Resource Independent Groups (RIGs)
  - Step 3: Compute the migration cost (migration time + wait time + impact on other RIGs) for each RIG
  - Step 4: Migrate the RIG with the minimum migration cost

Problem Constraints
- Only a single migration takes place for a single VM at any time
- After a migration starts, it cannot pause before completing the migration
- When a VM is migrating, resources are allocated in both the source and destination
- During migration, capacity constraints for CPU, memory, and disk must be satisfied.

Simulation Results
- Virtual specifications (CPU and memory) are taken from Amazon EC2 instances
- Bandwidth of virtual links are selected randomly between 250 and 750 Mbps

Conclusion
- VM migration sequencing is an important problem as it has a direct impact on
  - Resource utilization
  - Service performance
- Our solution CQNCR for VM Migration planning allows to
  - Avoid network congestion
  - Reduce total migration time by up to 25%
  - Reduce VM and VN downtime by up to 60%

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
3.3 Design and Management of DOT: A Distributed OpenFlow Testbed


With the growing adoption of Software Defined Networking (SDN), there is a compelling need for SD-N 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.
Emulating SDN

- An emulator for SDN can help in:
  - Testing and evaluating SDN-based control applications, policy enforcement platforms, monitoring frameworks, etc.
  - Gradual rollout of SDN-based solutions in production environments
- Mininet is the de facto standard SDN emulator

What Mininet does:

- Emulates the entire network in a single machine
- Uses Linux container to emulate end hosts

What Mininet cannot do:

- Cannot scale with network size and traffic volume
- Cannot emulate network services that require OS level isolation

Dot OpenFlow Testbed (DOT)

- Emulates an SDN network
- Distributes the emulated network across multiple machines
- Simulates link bandwidth and delay
- Supports both OS-level and full virtualization for emulating end hosts

Design Goals

- Scale to meet the requirements of the emulated topology by leveraging multiple physical machines
- Keep the distributed deployment hidden from users
- Provide resource guarantees for all emulated components (i.e., switches, links, and hosts)
- Allow emulation of a larger class of network services (e.g., managing middleboxes)

Resource Allocation

- Resource allocation problem is formulated as an ILP

Objective function:

\[
\text{Minimize } \alpha C^T + \beta C^E
\]

Translation cost, \(C^T\)

\[
C^T = \sum_{i \in N} \sum_{p \in R} \gamma_{i,p} y_{i,p}
\]

Operational cost, \(C^E\)

\[
C^E = \sum_{i \in N} \sum_{p \in R} \delta_{i,p} x_{i,p}
\]

- Capacity constraint

\[
\gamma_{i,p} + s_i \leq x_{i,p} + y_{i,p}, \quad \forall i \in N, \quad \forall p \in R
\]

- Bandwidth constraint

\[
\gamma_{i,p} + s_i \leq x_{i,p} + y_{i,p}, \quad \forall i \in N, \quad \forall p \in R
\]

- Delay constraint

\[
\gamma_{i,p} + s_i \leq x_{i,p} + y_{i,p}, \quad \forall i \in N, \quad \forall p \in R
\]

This formulation generalizes Multi-dimensional Bin Packing Problem \(\Rightarrow \text{NP}-hard\)

Resource Allocation Heuristic

- Select a switch \(i\) and assign it to a physical host \(p\)

Switch selection

- Select a switch \(i\) using

\[
R_i = \gamma_{i,p}^2 + \gamma_{i,s} + \gamma_{i,p}
\]

Where

- \(R_i^2\) → Degree ratio
- \(R_i^2\) → Resource ratio
- \(R_i^2\) → Neighbor ratio

- Host selection

- Select a physical host \(p\) for switch \(i\)

\[
F_p = 2 \gamma_{i,p}^2 + \delta_{i,p}^2
\]

Where

- \(F_p\) → Residual capacity ratio
- \(F_p\) → Locality ratio

- Repeat until all switches are reassigned or no embedding is possible

Evaluation

- Comparison to Mininet
  - Foreground traffic: UDP traffic at a rate of 1Gbps between C and S
  - Background traffic: 7 UDP client-server pairs are chosen randomly
- Performance of resource allocation heuristic
  - Four ISP topologies from RocketFuel repository
  - Our heuristic uses less physical resources (physical machines and link bandwidth) than First Fit approach

Conclusion

- DOT solves scalability problem of Mininet
- DOT hides distributed deployment of emulated network from SDN controller
- DOT provides opportunities to emulate a wider range of network services

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 NETGP334424-10
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.

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.

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)

IMPLEMENTATION EXAMPLE
- Centralized
- Beacon (1.0.4)
- Coordinated
- NOX (Verity)

Use Cases

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

- **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

- **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

### Security and Monitoring Application in OpenFlow

- **Need Packet Level Information**
  - 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, large k → only sample first m packets  
  - Suitable for applications such as traffic classification  
  - δ >0 → 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  
  - TCP/PSH 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 physical hosting infrastructure (e.g. data centers), there is an 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 this 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.
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 fluctuations.

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.
– Adapt VI embedding according to demand fluctuation.

Related Work

– Virtual Network Embedding: focus on static topologies only.
– Service Placement: Does not consider bandwidth requirement.
– Autonomic Service Provisioning: Focus on single data center scenario only.
– Research challenges:

  • How to define the VI request that enables dynamic scaling?
  • How to perform dynamic VI provisioning at runtime?

2 Request Specification

• Each request consists of multiple services \( J \).
• Each service is defined by a service graph \( S = (F, E) \).
• Demand originates from \( G \subseteq N \) in the physical topology.

– Service specifications:

  • Each service in \( N \) has resource requirement \( r^i \) for each resource \( i \) in \( C \).
  • Each server in \( N \) can be placed in locations \( R \subseteq N \).
  • Each link in \( F \) has a bandwidth requirement \( c^e \) for processing a single request of type \( e \).

– Delay requirements:

  • There is a set of paths \( P^e \) that have delay requirements.
  • Each link in \( F \) has an inter-server communication requirement.

– Inter-server communication requirement:

  • Bandwidth requirement \( B = a + b \) at \( \delta t \) seconds, where \( a \) is the total demand processed at \( \delta t \) and \( b \) is the total demand processed at \( \delta t \).

3 An Example

• We consider a simple web application with an authentication service.

4 Problem Formulation

• The goal of the dynamic VI scaling problem is to minimize the cost of the virtual network embedding problem.

\[
C = \sum_{i=1}^{n} \left( \sum_{e \in E} e^i \right) + \left( \sum_{e \in E} e^i \right)
\]

– Flow constraint:

\[
\sum_{e \in E} e^i = \sum_{e \in E} e^i \quad \forall r^i \in N
\]

– Delay constraints:

\[
\sum_{e \in E} e^i > \sum_{e \in E} e^i \quad \forall r^i \in N
\]

– Capacity constraint:

\[
\sum_{e \in E} e^i = \sum_{e \in E} e^i \quad \forall r^i \in N
\]

5 Solution Algorithm

• The key challenge is to find a trade-off between satisfying delay requirement and minimizing solution cost.

– Place servers based only on cost may not lead to a feasible solution.

– Place servers based on delay may lead to an high cost solution.

• Our approach is to try define an values for the trade-off factor \( \delta \) for each trade-off factor \( \delta \) can solve the assignment problem using greedy or multi-commodity flow algorithms and find \( \delta \) that produces the best solution.

6 Experiment Results

• We implemented our algorithm and compared it with two heuristics:
  – Min-cost heuristic: places each node-based solely on one source cost.
  – Min-distance heuristic: place each node to minimize total distance.

• Topology are randomly generated with demand fluctuations over time.

• The prob of a VI is determined by the size resource (total demand * revenue per request) minus embedding cost.

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:

  – Scale VIs across the 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 solution.
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.
**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 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. **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.

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

\[ \text{utility} = \sum_{t=1}^{T-1} \sum_{b \in B} \sum_{i \in U} \sum_{j \in \mathbb{B}_i} Q_{ij}(t) + V \cdot \sum_{t=1}^{T-1} \text{Cost}(t), \]

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

**Solution Algorithm**

Queue \( Q_{ij}(t) \) is kept for each user \( u_i \) at its associated base station.

Queues evolve according to the following queuing dynamic:

\[ Q_{ij}(t+1) = Q_{ij}(t) - \sum_{b \in \mathbb{B}_i} y_{ij}(t)w_{ib}(t) + w_{ij}(t), \]

where \( w_{ij}(t) \) is the arrival workload for user \( i \) and \( y_{ij}(t) \) is a binary variable that indicates activation of base station \( b \).

**Objective** is to minimize

\[ \limsup_{t \to \infty} \frac{1}{t} \sum_{r=1}^{t} \mathbb{E} \{ \text{Cost}(r) \}, \]

subject to network stability condition

\[ \limsup_{t \to \infty} \frac{1}{t} \sum_{r=1}^{t} \mathbb{E} \{ Q_{ij}(r) \} < \infty. \]

To solve the problem, the Lyapunov function \( L(t) \) is defined as

\[ L(t) = \sum_{b \in B} \frac{1}{2} \sum_{i \in U} \sum_{j \in \mathbb{B}_i} Q_{ij}(t)^2. \]

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

\[ \mathbb{E} \{ L(t+1) - L(t) | Q(t) \} + V \cdot \mathbb{E} \{ \sum_{t=1}^{T-1} \text{Cost}(t) \}, \]

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

**Model**

We consider downlink of an OFDMA-based cellular network, e.g., downlink in an LTE network. The system consists of the set \( B = \{ b_1, \ldots, b_N \} \) of BSs, the set \( U = \{ u_1, \ldots, u_M \} \) of users, and the \( H = \{ h_1, \ldots, h_T \} \) 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**.

The activation/deactivation decisions are made at the beginning of each time frame.

**Numerical Results**

A network of 25 BSs is considered. The behavior of GSA is studied in terms of energy cost and delay tradeoff. These parameters can be approximated via the number of active BSs and average queue sizes respectively. Each frame consists of 30 timeslots. Frames are divided into framesets of size 5. During even-numbered framesets (timeslots 0, 40, 100, 149, ...), user queues receive data while there is no data arrival in odd-numbered framesets.

By increasing \( V \), the average queue size increases and the average number of active BSs decreases.

**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 with temperature \( \Theta \) 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 \( \Theta \).

**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 NETGP399424-10.
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.

References


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 NETGP344424-10

Future Work

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

User Interface

- Creation of a new VDC:

- VDC Management:

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.

OpenStack

Web Interface

- VDC Requirements
- Status & Statistics

VDC Planner

- VDC Scheduler
- Monitoring & Measurement Module

Provisioning modules

OpenStack

Nova
Swift
Quantum
Glance-API
Cinder

VDC Planner DB

Web Interface

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.
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.

Throughput and Airtime Analysis

- Throughput of MS s at AP a is
  \[ T^a_s = \frac{P^a_{\text{coll}}}{P^a_{\text{idle}}} \cdot T\text{MP} \]
- Probability of successful transmission
  \[ P^a_{\text{succ},a} = \tau^a \prod_{\ell \notin a} (1 - \tau^\ell) = \frac{1}{1 + \tau^a} \]
- Consequently,\n  \[ T^a_{\text{succ},s} = \frac{\tau^a}{1 + \tau^a} \prod_{\ell \notin a} (1 + \tau^\ell) \]
- Airtime of MS s at AP a is
  \[ T^a_{\text{air},s} = \frac{P^a_{\text{coll}}}{P^a_{\text{idle}}} \cdot T\text{MP} + \frac{1}{\tau^a} \]
- Probability of collision
  \[ P^a_{\text{coll},a} = \tau^a \prod_{\ell \notin a} (1 - \tau^\ell) = \frac{1}{1 + \tau^a} \]
- Consequently,\n  \[ T^a_{\text{coll},s} = \frac{\tau^a}{1 + \tau^a} \prod_{\ell \notin a} (1 + \tau^\ell) \]

Numerical Results

WLAN Throughput vs Density

- Homogeneous Distribution
- Non-homogeneous Distribution
- Varying Load Balance

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

Association Control

- MS-AP association optimization problem
  \[
  \begin{align*}
  \max_{x, y, \tau} & \quad \sum_{s \in \mathcal{S}_a, a \in \mathcal{A}} P^a_{\text{coll}} (1 + x^a) - y^a, \quad \text{s.t.} \\
  \text{C11:} & \quad \sum_{s \in \mathcal{S}_a, a \in \mathcal{A}} x^a (1 + x^a) - y^a \geq y, \forall y \in \mathbb{I} \\
  \text{C12:} & \quad 0 \leq y^a, \forall y \in \mathcal{A} \\
  \text{MS-AP association through complementary geometric programming}
  \end{align*}
  \]

Conclusions

- It is shown that the 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, minimum airtime reservation for each ISP proposed algorithm improves total network throughput compared to the Max-SNR algorithm.

References


Acknowledgments

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.
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

Parsaei Fard, Mahsa Derakhshani, 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, $K_{rsv}^g$, is preserved for each slice; 2) Bandwidth-based Slices: minimum required bandwidth, $K_{rsv}^g$, is guaranteed for each slice.

SLOT 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 $P_{n,\text{max}}/\sigma = 20 \text{ dB}$.

ADMISSION CONTROL POLICY AND ITS PERFORMANCE

Due to channel fading, maximum power limitations and user 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 $P_{n,k} = x_{n,k}/w_{n,k}$ for the admission control policy.

ALGORITHM

- **Initialization**: Choose feasible values of $w^*(l = 0)$ and $P^*(l = 0)$, and set $l = 1$.
- **Step 1**: For fixed $P^*(l - 1)$, solve subcarrier assignment subproblem to derive $w^*(l)$.
- **Step 2**: For fixed $w^*(l)$, solve power allocation problem to derive $P^*(l)$.
- When $||P(l) - P(l - 1)|| \leq \epsilon$, 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 emphasizes the importance of admission control.
- Fig. 5 demonstrates the performance of admission control vs the location of users and $R_{rsv}^g$.

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


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 NETCFP74424-10.
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.


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

Prabhat Kumar Tiwary, Dr. Quang Dung Ho, Tho Le-Ngoc

Abstract
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 are non-trivial. Moreover, supporting service-oriented architecture and software-defined paradigm adds additional management complexity. This article first discusses the main existing and foreshadowed management issues. Second, the management requirements for virtualized and software-defined wireless infrastructures are explored. Last, a draft initial management framework along with a set of management specifications have been put forward.

Management Issues

a) Tenant-facing management issues:
- Different types of tenants: different service needs, requirements and control capabilities
- Two main tiers of tenants: infrastructure-level tenants and service-level tenants
- The issue is to devise schemes for management interfaces so that they can abstract and provide varying degrees of control depending on the type of tenant within the same management framework.
- Furthermore, allowing tenants to have tailored management modules on different levels of autonomy with the same framework is also a design issue.

b) Resource-facing management issue:
- A large variety of wireless technologies and standards: range of possible virtual resources (obtained from different virtualization perspectives) - varying range and granularity of control and management (C&M) capabilities
- The issue is to architect the ways to provide flexibility and extensibility in a framework such that the enabling/disabling of C&M capabilities become possible without changing the entire framework at a whole.

Management Requirements

- Requirements for SDI paradigm in wireless infrastructure
- Extensibility requirement
- Scalability requirement
- Typeology requirement
- Location sensitivity
- Resource management requirements
- Multi-tenant support
- Multi-tenant in-and-out support
- Requirements for SOA model in wireless infrastructure

Management Specifications

a) Hierarchy/layering in the management framework:
- Two basic layers: Infrastructure manager or C&M hypervisor, and Tenant Manager
- Hierarchy is the key to supporting different tenant's management capabilities and requirements, per tenant tailored management modules, and high-level SDI APIs.

b) Specification for management isolation:
- Possible due to the hierarchy in the framework
- Tenant manager only responsible for per-tenant high-level instructions and command passings
- Infrastructure manager incorporates isolation through passing, interpretation, conflict management and resolution.

c) Specification for modularity:
- Two kinds of modules in the infrastructure manager: Core management modules and pluggable management functions:
- Modularity makes it possible to extend the C&M functionalities to support various virtualization perspectives.
- Management functions can be added/removed in the framework in a plug-and-play and unpluggability fashion.

d) Specifications for diverse autonomy:
- Autonomous modules: Semi-autonomous modules: Non-autonomous modules

- Centralized Management Topology
- Distributed Management Topology

- Fig. 2: Requirements of a Management Framework for Virtualized and Software-Defined Wireless Infrastructures
- Fig. 3: An Initial Draft Management Framework (OpenMan) for Virtualized and Software-Defined Wireless Infrastructures
- Fig. 4: Diverse Autonomy of Management Modules
- Fig. 5: Direct and Interception modes
- Fig. 6: Management Topology Proposals for Different Wireless Technologies
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.
Aurora: A Virtualization and Software-Defined Infrastructure Framework for Wireless Networks

From Prabhat Kumar Tiwary, Kevin Han, Hoai Phuoc Truong, Dr. Quang Dung Ho, Tho Le-Ngoc at McGill
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.
Introduction

- Working group: Theme 4 (Integrated Wireless/Optical Access)
- Third year objectives: Develop and demonstrate full duplex wireless WiFi/LTE and digital transmission over fiber
- Current Proposition: RSOA-based carrier remodulation for WDM-PON networks. We implement a full-duplex WiFi analog with digital downlink radio-over-fiber system for FTTx applications.

Architecture

- Challenges solved and to solve:
  - Scalable architecture, low cost at the remote antenna, Wireless and optical compatibility
  - Centralized DSP, and power-hungry components
  - Support multiple services/standards
  - Compatible with SISO/MIMO schemes
  - Flexibility for FFTA, FFTH, and more
  - Extend the RoF link as much as possible

Setup & Experiment

- We use the same optical wavelength to send WiFi analog signal on different RF band with OOK downlink.
- The modulation performance dependence of the RSOA is investigated.
- Multiservice transmission performance is demonstrated.
- We evaluate performances of UL/DL for different modulation formats: QPSK, 16QAM and 64QAM OFDM over different fiber length.

Full duplex WiFi analog signal transmission in FDD mode over 20 km (64QAM), 30 km (16QAM) and 40 km (QPSK) within FEC performance regions at the presence of downlink error free OOK signal.

- Uplink: Dashed lines
- Downlink: Solid lines

RSOA Impact

Greater bias current gives better EVM inside the grey zone. The increased ASE noise explains the bad EVM outside the grey zone at 80mA.

The grey zone shows increased tolerance to larger RF swing voltage biased at 80mA as opposed to 50mA, i.e. increased dynamic range.
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.
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
Intensive Care Unit Waiting Room

Flexible and Dynamic Setup and Management of a WLAN with Aurora

Step 1
Change SSID of slices in the SAVI Wireless Network

Step 2
Add access slice on Triage with SSID NurseIntern

Step 3
Move the NurseIntern slice setup to ICU

Step 4
Create a new slice on Triage

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
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
- Partial Reconfiguration – the ability to reconfigure a section of an FPGA
  - At runtime
  - Without affecting other running circuits
- Challenge - How to do this?

Goals: Virtualize FPGA Resources
- Goal: virtualize FPGA resources to enable cloud-based hardware acceleration

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

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 floating at variable rates
- Much higher throughput!

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
- 1 perf with and without flow redirection to VFR
- Virtually no difference in throughput!

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

OpenStack Nova
- Common API
- Driver
- Resource-Agnostic Management

Architectures
- Field Programmable Gate Array
  - Tiled array of programmable logic blocks, interconnected by programmable routing architecture
  - Hard-block RAMs, DSP Blocks (multipliers), high-speed transceivers
  - Results: 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

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

Common API
- Driver
- Resource-Agnostic Management

Drivers
- Resource-Specific Management

Resources
- Resource-Specific Management

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
In this work, we leveraged SAVI to provide a network solution to enhance VoIP media quality by using a converged and centralized resource management framework. We will demonstrate that with a reconfiguration capability of the SAVI platform, the end-to-end real-time VoIP video call can be observed in a higher level of media fluidity.

Introduction
VoIP services are becoming a staple of network packet loss, and also require large end-to-end bandwidth for HD media flows between ends. To this effect, we propose a new dynamic mechanism to repair the video fluidity of the call session. OpenFlow technology provides a solution.

1. Forwarding logic — Controller.
2. Forwarding processing — OpenFlow-enabled switches.

Traditional IP networks do not support a fast reconfiguration to resolve such a problem in real-time. Our approach leverages a real-time manner.

SAVI testbed has converged with the virtual network functions to configure the SAVI-enabled network. In this work, we demonstrate the capability of SAVI SDI to provide an end-to-end media flow with congestion-free usage of each physical and virtual servers and switches.

Determinants of Media Quality
The media quality of VoIP call sessions is determined by:
1. Codec — efficiency parameter, image capture size, pixel, and frame rate — fluidity
2. Network condition — in particular, VoIP session bandwidth usage, latency.

IMS Network Embedding
The IMS network segments into two parts:
1. Signalling plane - the flows of SIP/2 messages that are exchanged between UE and IMS domain.
2. Media plane — the flows of audio/video RTP packets sent between UE and IMS domain.

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

VoIP Service on SAVI Testbed
We have developed a SAVI platform-based VoIP residential service leveraging OpenFlow. The platform uses a controller that manages all switches, virtual servers, and physical servers. The SAVI platform can be manually configured with OpenFlow-enabled network elements.

The SAVI testbed has provided an application platform for the VoIP services. The controller monitors the usage statistics of each physical and virtual servers and switches. The platform provides a real-time enhancement of VoIP services.

Table 1: Network Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet loss rate</td>
<td>≤ 1%</td>
</tr>
<tr>
<td>Packet loss</td>
<td>≤ 1%</td>
</tr>
<tr>
<td>Latency (ms)</td>
<td>170 ≤ 20 ≤ 170</td>
</tr>
<tr>
<td>Bitrate (kbps)</td>
<td>1280 × 720 = 917kbps</td>
</tr>
</tbody>
</table>

Table 2: Network Testbed

<table>
<thead>
<tr>
<th>Testbed</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAVI</td>
<td>VoIP session</td>
</tr>
<tr>
<td>VoIP session</td>
<td>1080p</td>
</tr>
<tr>
<td>Bitrate (kbps)</td>
<td>1080p</td>
</tr>
<tr>
<td>Latency (ms)</td>
<td>170 ≤ 20 ≤ 170</td>
</tr>
<tr>
<td>Frame rate (fps)</td>
<td>30</td>
</tr>
<tr>
<td>Frame size (pixels)</td>
<td>1280 × 720</td>
</tr>
</tbody>
</table>

Table 3: Summary of test results

<table>
<thead>
<tr>
<th>Testbed</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAVI</td>
<td>VoIP session</td>
</tr>
<tr>
<td>VoIP session</td>
<td>1080p</td>
</tr>
<tr>
<td>Bitrate (kbps)</td>
<td>1080p</td>
</tr>
<tr>
<td>Latency (ms)</td>
<td>170 ≤ 20 ≤ 170</td>
</tr>
<tr>
<td>Frame rate (fps)</td>
<td>30</td>
</tr>
<tr>
<td>Frame size (pixels)</td>
<td>1280 × 720</td>
</tr>
</tbody>
</table>

Conclusion
In this work, we demonstrated the capability of SAVI SDI platform through a multimedia application - VoIP service. The SAVI testbed has converged with the virtual network functions to configure and manage network elements and provide a real-time quality enhancement solution to VoIP services adversely affected by congestion in the network.

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 NETSP-064242-10.
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

Resource Types
Cloud resources that are describable by a Heat template include:

- OS::Nova::Server
- OS::Nova::KeyPair
- OS::Neutron::Network
- OS::Neutron::FloatingIP
- OS::Cinder::Volume
- OS::Neutron::SecurityGroup
- OS::Swift::Container

Key Advantages
- Reduce the complexities of application deployment and management at the SAVI Testbed.
- Provide a high-level Domain Specific Language (DSL) and APIs.
- Tightly integrate with other Testbed's services.
- 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.

Introduction
- Smart Applications on Virtual Infrastructure (SAVI) is a Canadian research project that explores the benefits of a hierarchical cloud infrastructure platform.
- 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.

Figure 5.3: An Orchestration Service of SAVI Testbed; A Heat Approach from Hongbin 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.
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 platform.
- 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 API.
- By integrating Sahara into SAVI Testbed, we allow 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
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

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 distributor
- 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.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.CO - 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 / Subscriber 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 package with OpenBTS.
- Asterisk: It is the "standard" OpenPBX (Private Branch Exchange), acting as the telephone switch. It is the most documented option, yet the most difficult (vs. FreeSwitch and Yate).

Acknowledgements

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

References

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’s Alarm feature
- A monitoring system that allows users 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 users 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

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.

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

Introduction

Group communication is widely used in modern communication networks. Multicast lends itself naturally to these communication patterns. By 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 BGPv6, GFS, and clustered application servers. The former multicast techniques on SAVI network are not efficient and robust. The implementation of this multicast mechanism will help avoid 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 shortest tree is an NP-hard problem, heuristic algorithms have been proposed. (1) provides an algorithmic 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 linear path concentration. (3) proposed an SDN-based system that enables multicast in commodity switches without the data center. 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 paper, 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 network topology and the shortest path from any node to any destination node. The multicast mechanism implements the shortest path algorithm.
3. The multicast mechanism implements the shortest path algorithm 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 using bandwidth and minimizing transmission delay.

Abstract

Group communication is widely used in modern communication networks. Since the problem of finding the shortest tree is an NP-hard problem, heuristic algorithms have been proposed. (1) proposes an algorithmic 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 linear path concentration. (3) proposed an SDN-based system that enables multicast in commodity switches without the data center. 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.

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)

1. Multicast(edge network, src node, dst nodes, threshold)
2. Command Generator
3. Takes the input from the file
4. Multicast-tree = [src node]
5. multicast-tree.append(candidate path)
6. for each path in path list
7. delay = total transmission delay of path
8. multicast-tree.append(path)
9. end for
10. end

Conclusions

- We propose an efficient multicast mechanism and implement it 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

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.

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

Simulation Results

- Video Clip Profile: Video Stream Traffic
- Video Clip Profile (No traffic control):
- Video Clip Profile (With OF/OVS traffic control):

Future Work

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

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


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.
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