Traffic Optimization in Multi-Layered WANs using SDN

Henrique Rodrigues[1], Inder Monga[2], Abhinava Sadasivarao[3], Sharfuddin Syed[3],
Chin Guok[2], Eric Pouyoul[2], Chris Liou[3], Tajana Rosing[1]

[1]University of California, San Diego
La Jolla, CA, USA

[2]Energy Sciences Networks
Berkeley, CA, USA

[3]Infinera Corporation
Sunnyvale, CA, USA

***Abstract* — Wide area networks (WAN) forward traffic through a mix of packet and optical data planes, composed by a variety of devices from different vendors. Multiple forwarding technologies and encapsulation methods are used for each data plane (e.g. IP, MPLS, ATM, SONET, Wavelength Switching). Despite standards defined, the control planes of these devices are usually not interoperable, and different technologies are used to manage each forwarding segment independently (e.g. OpenFlow, TL-1, GMPLS). The result is lack of coordination between layers and inefficient resource usage. In this paper we discuss the design and implementation of a system that uses unmodified OpenFlow to optimize network utilization across layers, enabling practical bandwidth virtualization. We discuss strategies for scalable traffic monitoring and to minimize losses on route updates across layers. A prototype of the system was built using a traditional circuit reservation application and an unmodified SDN controller, and its evaluation was performed on a multi-vendor testbed.**

Keywords—Wide Area Network; Software Defined Network; OpenFlow; Multi-layer; Virtual Network; Traffic Engineering

I. INTRODUCTION

Increased adoption of cloud computing and global scale distributed systems propel the growth of network demand across the globe. Predictions are that the Internet will grow significantly in the next few years, carrying Zettabytes of data in 2017 [1]. Wide area networks (WAN) are at the core of the global inter-network infrastructure, delivering several terabits per second across thousands of high bandwidth capacity links. These networks support a wide range of Internet services, and are critical to the reliability and performance of the Internet.

The exodus of traditional applications to data centers and the rise of large "Big Data" datasets contribute not only to traffic growth but also to variable inter-data center traffic demands. Traditional WAN management systems handle demand variations by routing traffic through multiple network paths, using solutions such as MPLS-TE, B4 [2] and SWAN [3]. Multiple Optical Network Elements (NE) compose the underlying transport infrastructure for these paths, as they offer cost-effective high bandwidth forwarding for interconnecting globally distributed routers. The problem, however, is that these optical nodes are often invisible to traffic engineering (TE) systems. As a result, optimizations explored by routers or systems such as B4 and SWAN are limited to a logical portion of the network, which is mostly static. If the demand at the IP layer grows beyond its limits, new optical paths need to be created, and this might involve provisioning actions beyond the capabilities of current systems. Furthermore, variable demands under the current practice of static bandwidth allocation based on peak usage is likely to result in siginificant resource wastage as optical speeds evolve past 100Gbps [12].

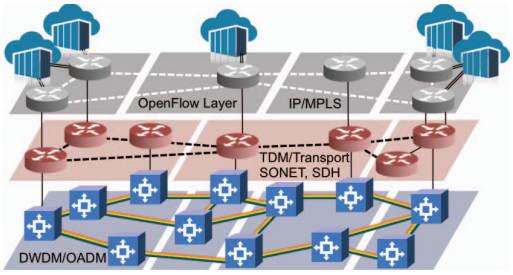

Figure 1: Segregated nature of WANs, with network devices grouped in layers managed by their forwarding technology.

Figure 1 depicts a segregated WAN, with multiple network nodes spread across 3 management layers. The actual number of independently managed layers can vary, depending not only on forwarding technology, but also on the lack of interoperability between systems orchestrating each segment. In some cases, layers are even induced by carrier-customer relationships, with some carriers also having other carriers as customers [4]. Management segregation can also be present in the same institution, with different layers operated by different engineering teams [5]. This motivates WAN providers to rethink how these networks are managed [4]: instead of focusing on traffic engineering at each layer independently, orchestration across layers becomes important to ensure low operational overheads as well as high utilization of resources.

In this paper we present the design and implementation of OSCARS-TE, a multi-layered traffic management application designed to provide bandwidth on demand and ease TE across segregated network data planes. We take advantage of the fact that Optical Transport Elements and Transparent Optical Switches are improving their programmability [9], and suggest the use of a unified control plane (UCP) based on OpenFlow (OF) to manage both packet and optical transport nodes. Unlike previous approaches, which propose OF extensions to handle particularities of the optical layer, OSCARS-TE does not need any changes to the protocol. Instead, we propose a new version of the Open Transport Switch (OTS) [9], which maps the particularities of lower layer elements to the standard OF protocol, allowing OSCARS-TE to manage traffic allocation using SDN application logic.

The use of unmodified OpenFlow allows network operators to plug in traditional Software Defined Network (SDN) controllers, re-using TE policies and management systems for multiple forwarding layers. With global view of the network across multiple layers and simple control interface, OSCARS-TE can provide elastic management of available bandwidth to eliminate performance problems. We discuss the challenges

978-1-4799-5860-3/14 $31.00 © 2014 IEEE
DOI 10.1109/HOTI.2014.23

and design decisions to perform multi-layered network optimization, and evaluate an initial prototype of OSCARS-TE in a multi-layer network composed by packet and optical transport devices.

II. Background and Motivation

The evolution of optical network speeds, from current 100Gbps towards Terabits per second in a few years, highlights the need for flexible bandwidth allocation in the optical plane. The traditional practice of static end-to-end resource allocation, facing variable inter-data center WAN traffic demands [11], can result in significant wastage of resources. Enabling technologies exist in both the optical domain [12], as well as in the packet switched domain [2][3]. Bridging the gap between these two domains can enable application centric orchestration of large bandwidth pipes, leading to cost-effective use of the network infrastrcutre. The problem is that the legacy layered infrastructure of WANs segregates resource provisioning and management, imposing a few barriers for dynamic orchestration across domains. Here we briefly describe the reasons for curent network segregation.

1. Infrastructure

The lower layers of WANs are composed of Reconfigurable Optical Add-Drop Multiplexers (ROADMs) interconnected by optical fibers, which are preferable over copper for long distance transmissions due to their low propagation loss. They carry digital information encoded into different optical wavelengths, with ROADMs routing wavelengths by filtering and directing them to different fibers. These devices have no knowledge of the actual application-level data streams modulated into optical signals (i.e. IP routing, application demands), and provide static amount of network resources between nodes regardless of how much useful data is transmitted. Wavelength-Division Multiplexing (WDM) enables cost-effective high bandwidth by combining wavelengths into a single fiber.

Above we have the Optical Transport Layer, which is also referred to as digital wrapper or optical channel wrapper. Here network resources are provisioned as sets of optical cross connects (XCON), defining optical circuits that interconnect two other higher layer devices. Optical signals can be processed based on electronic frames or optically, when done at higher speeds. This layer allows further division of bandwidth available in single wavelengths with Time-Division Multiplexing (TDM), usually in incremental steps from a basic unit such as ODU0 [15]. This provides higher flexibility on resource allocation compared to the physical layer.

Packet forwarding using IP/MPLS is at the top of the infrastructure, being responsible for most of the "dynamic" network intelligence: routing, TE, and fault tolerance. The detailed packet switched nature of this layer gives it the highest level of flexibility; enabling traffic management at the level of inter-application flows.

2. Network Management

Due to their different technology and resource provisioning, each layer has its own management interface, which are usually non-interoperable [4][13]. The disparity between layers comes from their distinct purposes as they evolved over time. Transport networks and long distance traffic forwarding protocols evolved from telecommunication networks, with goals such as reliable delivery and strict latency requirements, both essential for reconstructing a live voice stream. This allows efficient forwarding at high speeds with shallow buffers, as switching is relatively simple and resources are often reserved beforehand. In contrast, data networks were born as a best-effort service, designed to tolerate delays, resource contention and variable availability. Their unstructured nature also requires features such as forwarding loops prevention and modest sized buffers, which absorb demand variations.

Bandwidth provisioning multi-layered networks is usually done in multiple steps (e.g. RSVP-TE, LMP, TDM circuits, IP rules at routers) and involves multiple management interfaces (e.g. TL-1, NETCONF, SNMP), as well as possible manual provisioning needed for various devices [4]. TE and bandwidth allocation in the packet layer is done *after* this step, taking as input a static network graph. The need for bandwidth allocation and orchestration across multiple devices has motivated the design of unified control planes (UCP). A well-known solution for the optical domain is the Generalized Multi-protocol Label Switching (GMPLS) protocol. However, despite addressing some of these problems, GMPLS-based control planes have not been widely deployed for multiple reasons [13]. Most notably, carriers indicate strong preference for a centralized solution [14].

A promising paradigm, successful in the packet layer, is Software Defined Networking (SDN). It proposes a complete separation of data and control planes, giving total control over data plane forwarding decisions to a remote application, or a SDN controller. The advantage is that network orchestration can be done with global knowledge of the network state, enabling globally optimal routing and TE decisions [1][3]. A common control interface, such as the OpenFlow protocol, is key to the success of SDN, providing a generic abstraction for packet switched data planes. This enables coordination of devices from multiple vendors using the same interface, eliminating costs of integration development and vendor-induced network segregation ("vendor-islands").

OpenFlow extensions for optical data planes are explored in previous works [5][6][7][9]. However, proposed changes usually conflict with the generic OpenFlow control interfaces, slowing down standardization and also their adoption by different vendors. Furthermore, the proliferation of vendor extensions induces more segregation, resulting in reduced interoperability and extra development costs for integration.

The inexistence of a unified control protocol is not the only limiting factor for practical cross-layer orchestration. Traffic engineering and route updates in multi-layered networks are also different. One problem is the resource provisioning time of the different layers. Packet switches are able to change its data plane instantaneously, enabling multiple concurrent route updates in short timescales [2][3]. Traditional ROADM and mechanical-based optical switches were not designed for such needs, and offer provisioning time orders of magnitude slower than electronic based data planes [16]. Even when both

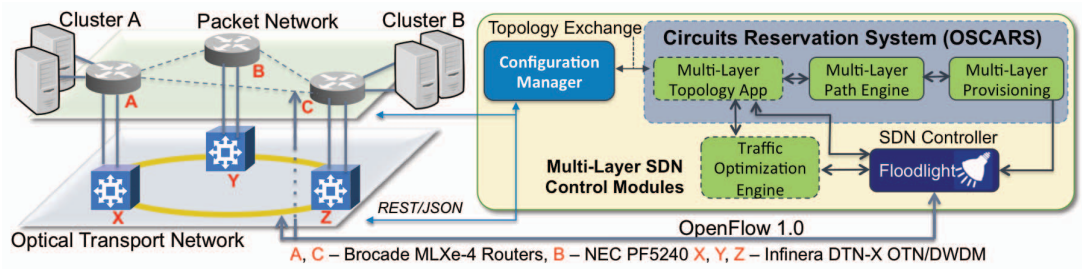

Figure 2: Experimental environment, multi-layer application modules and interactions between application and devices

optical and packet layers operate electronically, the interaction of different forwarding technologies (such as Ethernet over SONET or OTN) can induce link availability delays. Therefore, careful handling of route updates across layers is necessary to avoid performance degradation. We describe how we solve this problem using planned multi-layer route updates in §III and quantify the performance degradation in §IV.

The absence of a complete multi-layer UCP to orchestrate segregated data planes motivated us to design and implement OSCARS-TE, a system that enables practical elastic bandwidth provisioning for multi-layer WANs. We also designed a lightweight version of the Open Transport Switch [9] that allows resource provisioning of optical transport nodes using standard OpenFlow 1.0 without modifications.

III. Architecture and Implementation

The system design borrows multiple components from the On-demand Secure Circuits and Reservation System (OSCARS) [11], an open source circuit provisioning software developed by ESnet and partners. We extended the original application to build OSCARS-TE, adding dynamic traffic monitoring, extensible topology management and elastic resource provisioning for effective cross layer resource optimization and orchestration. The architecture of the system is depicted in Figure 2, with standard Floodlight as our current SDN controller. However, before describing OSCARS-TE, we first briefly discuss how optical network elements can be virtualized and managed using unmodified OpenFlow.

1. *Open Transport Switch using Standard OpenFlow*

The common extensions that enable OF use for optical network elements usually introduce new control messages to indicate optical channel (OC) line rates and network element identifications [5][9]. Despite requiring SDN controller modifications, this pushes unnecessary responsibilities such as timeslots availability tracking – which are usually lower layer specific information – to the SDN controller.

Another approach is to map optical wavelengths into virtual ports, and perform wavelength routing at the OF controller [17]. The solution, however, is restricted to physical layer ROADMs and devices that operate at wavelength granularities, excluding low granularity TDM switching and newer higher speed optical technologies such as Nyquist DWDM (super channels) from the set of possible interconnections.

Practical packet-optical integration requires a more general approach for virtualization of optical layer. It should abstract

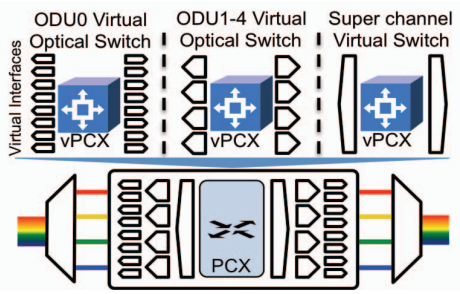

Figure 3: Optical Switch Virtualization using OTS: possible cross connections with matching ODU are exposed as a single virtual switch with a virtual PCX.

optical layer limitations and details, such as wavelengths, modulation formats vs. optical reach and/or timeslots, providing a simple interface that is better understood by TE systems.

In our new version of OTS, we virtualize all possible cross connections of optical network switches into virtual ports. All ports of same switching granularity (e.g. ODU0) are grouped into a virtual optical node, as shown in Figure 3. The switch exposes virtual nodes as a full cross bar switch with a virtual photonic cross-connect (PXC) of a specific bandwidth, which is the actual resource unit understood by applications. This gives the SDN controller a clear and simple view of the possible routes for traffic flows of a particular demand, independent of wavelengths, line rates and/or time slots. The abstraction, allows resource provisioning using traditional OF, as SDN controllers already support such abstraction with the *in_port* field in the OF match and the primitive *output* action. We direct the reader to [9] for an extensive discussion on the implementation of OTS. Our modifications affect only the exported interface, but uses similar architecture.

The abstraction can be applied to other optical switches, such as ROADMs, making the OTS implementation general enough to address optical networking elements operating in different layers. It also permits resource partitioning to different SDN controllers, similar to the service provided by FlowVisor, by splitting virtual interfaces between additional virtual nodes of same resource granularity. However, as in FlowVisor, this task is not addressed by OpenFlow, but accomplished by an out-of-band protocol like OF_CONFIG.

2. *System Overview*

The high level algorithm of OSCARS-TE can be described with three simple steps that obey the following control loop:

1. Monitor load on network devices every Ts seconds. In case of a network event, the level of detail in network monitoring is increased on nodes where the event started, in an attempt to determine its cause.
2. OSCARS-TE looks for possible optimizations, either by providing extra resources or separating traffic flows whose interaction can lead to poor network performance.
3. Once the optimization decision is determined, an optimized cross layer path is requested to the OSCARS Path Computation Engine (PCE). A route update plan, with an ordered sequence of node updates, is computed and executed to update the network state.

3. Automated Topology Discovery

One of the most important tasks performed by a SDN controller is to keep track of the network topology. Most controller tasks, such as routing, provisioning and fault recovery, need topology information. This information allows prompt network reconfiguration upon a failure with globally optimal routing decisions. This is also an advantage of SDN over distributed network control, which usually needs various steps and coordination messages to re-adapt the network state.

One of the challenges in implementing a multi-layered SDN application is that there is no standard, generic way to build a multi-layered topology graph. At the packet layer, LLDP works well to find (logical) links between switches. However, when it comes to optical interconnections (or the packet/optical edges), any L2 mechanism fail. These packets are transparent to optical nodes, leaving manual administrator input - possibly aided by proprietary tools - as the only option.

We use multiple sources to build a multi-layered topology graph. In the first step, we identify the connected packet switched components of the network using LLDP. Once this graph is constructed, the controller can scan all possible paths exported by the OTS, cross connecting active ports with unknown ends in the optical domain. A successful path interconnecting two packet switches generates a *link state change* OF message, revealing the physical links in the topology graph.

This method works for modest-size sections of the network, but it is not scalable. Therefore, the system can also accept vendor-specific topology information for the optical domain, and join the topology sources (both the dynamically inferred and the statically provided) together to create a full topology graph.

4. Scalable Traffic Monitoring

The main use for bandwidth on demand in the optical layer has been fast restoration of connectivity in case of failures. Other uses, such as rapid provisioning of bandwidth to application-level services have been a challenge, because while the optical layer has the resources, it lacks application level information, whereas in the packet layer, the situation is the opposite. We tried to break this gap with OSCARS-TE by collecting network-wide statistics and accomplishing rapid provisioning based on application demands.

However, monitoring all application flows in the network from a central controller is not scalable both to the server aggregating traffic measurements as well as for the constant polling of switch information. On a heavily loaded switch, querying and transferring all flow level statistics might even affect the performance of the control network. Even so, application level traffic is necessary to break the gap between the traditional static optical bandwidth allocation and elastic bandwidth provisioning to meet application demands.

To overcome this dilemma, we designed a scalable traffic-monitoring module that mitigates the need to keep constant application level flow information. We mitigate the problem by employing multiple network monitoring methods with distinct granularities of traffic information.

OTS can export virtual interface statistics using OpenFlow. Packet devices also offer coarse-grained interface information through OpenFlow or SMNP counters. Fine-grained traffic statistics can be retrieved via standard packet sampling. Multiple packet switches support this feature, and export the information using protocols such as sFlow or NetFlow.

We adopt a split model for network monitoring that leverage multiple solutions. OSCARS-TE restricts periodic monitoring probes to coarse-grained statistics, reducing data collection for periods of inactivity. We keep a moving average counter for each monitored point to alleviate small short-lived bursts of traffic. Case the processing of this information points to undesirable network events, such as packet drops and/or traffic patterns that might indicate a bottleneck, the system changes the data collection method to retrieve more fine-grained information. Such change in monitoring patterns, however, is constrained to the sections of the network involved in the network event. This enables the identification of application flows that are contributing to spot network congestion or that might require extra bandwidth.

In this initial implementation of OSCARS-TE, we use OpenFlow and sFlow, when available, to collect traffic statistics for interfaces and flows. Packet sampling can replace the statistics collection of OpenFlow, providing similar flow level information without the need for in hardware statistics counters. Some OpenFlow versions that support group entries can also be used for packet sampling, eliminating the need for sFlow support. We do not use such feature.

5. Demand inference and Elastic Bandwidth Allocation

Traditional WAN management systems, such as OSCARS, perform point-to-point resource reservation based on user-supplied demand. However, users might not have a complete understanding of application needs, and static bandwidth allocations lead to inefficient resource utilization when applications have variable demands.

Similar to previous systems, OSCARS-TE takes as input user-supplied demand information as an indication of the amount of resources to be reserved. However, it can manage the resources statically or dynamically, with elastic resource management. In the former, the system works as before, with static guaranteed bandwidth offering predictable network performance. The later avoids underutilization. In this case re sources are reserved only as a form of admission control. However, only the minimum amount of resources is provisioned for connectivity and small flows (i.e. ODU0/2 connec-

tions). The system then dynamically adjusts resource provision according to measured demands.

For elastic bandwidth allocation, OSCARS-TE processes the monitored traffic information looking for two types of events, which we refer to as ***balanced demand growth*** and ***unconstrained imbalance growth***. The first happens due to regular demand growth, usually as result of increasing number of applications sharing the allocated resources with each of them having approximately the same network demand. In this case, there is only one solution: provide extra bandwidth. This can be done as the measured demand exceeds a certain established threshold, such as 70% of capacity.

The second event is less obvious: moving large amounts of data between data centers can easily make a single application fill a multi-gigabyte link. This is the case in WANs such as ESnet, which carry bursts of large scientific data transfers with regular traffic through the same path [11]. When links carry such a mix of small intermittent flows from a few applications and high demand flows coming from other application collocated in the same site, behind an access router (i.e. in the same cluster, data center, research center, etc.), contention for network resources can cause performance degradation, and consequently, low link utilization. The interaction between these flows triggers the TCP congestion control for all flows. In an attempt to maintain fairness between flows, larger flows must reduce their bandwidth to match the bandwidth attributed to small flows. If the smaller flows vanish after the resources were equally shared, the remaining flows will start probing for the spare bandwidth incrementally, leaving some resources sub-utilized. If small flows are intermittent, this could be repeated over and over again.

We identify this event by correlating packet drops with low link utilization. Whenever unconstrained imbalance happens due to link contention between big and small flows, packet drops will be present in the monitored data, but the throughput on the provisioned circuit will not be maximized, due to constant TCP back offs. Increasing link capacity does not solve this problem, because load imbalance will still be present. Instead, we adopt another solution: split the groomed traffic in two circuits. We temporarily allocate a circuit with higher capacity for the large data transfer, which can then flow without interference from the small intermittent flows.

Application level flow admission, and different TCP congestion control algorithms can help prevent this problem [1][3]. However, some WAN operators do not have control over software running beyond the network edge. For ESnet in particular, the distributed management scenario of research laboratories makes it nearly impossible to have all servers using, for example, the same TCP variant.

6. Multi-layer aware topology update

Topology updates can result in performance degradation when it involves re-routing traffic through multiple network nodes [18]. Congestion-free topology updates for the packet layer was proposed before [3]. In a multi-layered network, the provisioning time of different layers can also induce performance degradation, because optical layer reconfigurations usually cause link interruptions [16].

Before offloading a flow, OSCARS-TE computes an offloading plan. The plan ensures minimum packet loss during route updates. Instead of setting up a path in sequence, hop-by-hop, the plan orders the updates such that lower sections of the network have higher priority, and end-points have the lowest priority. Whenever a route update in the optical domain is made, OSCARS-TE creates an event listener for link state changes of ports in the packet layer that are part of the path. The collection of these event listeners related to a topology update acts as a barrier for the progression of the update, which is only trespassed when a stable connection is detected in the packet layer. This avoids creating a black hole in the optical layer, which would drop all packets flowing through it, slowing down or even stopping the flows being offloaded. The updates on the packet layer are also done in a careful order to prevent performance degradation, as described in [18].

7. Differentiation of Node Capabilities

When performing multi-layered topology changes, it is important to distinguish nodes in different layers. Even though OF-enabled packet switches can be controlled using the same protocol, they usually have multiple differences that can prevent seamless integration. Examples are IEEE 802.1Q VLAN tag restrictions, flow table capacity and flow match fields. In terms of VLANs, switches might deny the use of some VLAN tags, reserving them for management and proprietary protocols. Flow table capacity is often different between devices [10]. Some vendors even allow a single device to be configured in different modes to explore hardware tradeoffs, such as increasing the flow table capacity while having a less detailed flow matching. The traditional practice is to minimize flow table usage to avoid unexpected problems [3].

OpenFlow provides feature request and feature response messages to specify what a device can do. However, it is not the complete solution. The protocol lacks a more general and flexible specification to distinguish node and interface *capabilities*. One possible capability that can be leveraged for multi-layer network management is switching speed (i.e. picoseconds for packets/flows and millisecond to seconds for optical/virtual interfaces). Understanding the limitations of a network device (such as the ability to switch traffic based on Ports,VLAN/L2,3,4 headers for example) and how it will react when a request is issued (how long it takes to have a XCON ready for example) can aid traffic management decisions.

To overcome this limitation until the protocol is mature and generic enough, we use an external database that is indexed by an OF Datapath ID (DPID), and contents specifying node capabilities. In terms of packet and optical capability distinctions, an obvious difference is optical elements can only switch traffic based on cross connects. As described before, this can be translated to *in_port* and output action in OpenFlow terminology. A not so obvious difference is that optical devices can have only one flow rule per interface, i.e. once an *in_port* match is instantiated for a port, all the bandwidth available for that port is allocated regardless of the required bandwidth. As OSCARS-TE keeps track of reserved bandwidth, it is important to decide where bandwidth would go at

path computation time, and how allocations affect the remaining bandwidth. To deal with this issue, we used the concept of allocation granularity from OSCARS, which dictates the minimum bandwidth to be allocated when a flow uses virtual interfaces. We also leverage OSCARS hierarchical Path Computation Engine [11] to find paths suitable for user-supplied requests that satisfy network restrictions.

IV. EVALUATION

Our experiments use the testbed depicted in Figure 2. We have one Brocade MLXe router, one NEC PF5240 switch and 3 Infinera DTN-X OTN/DWDM, interconnected as shown in the picture, and 4 servers. All servers have CentOS 6.4 with Linux 2.6.32. Two servers are used to run OSCARS-TE and Floodlight, and other two are used to send traffic through the network. Because we have only two hosts creating traffic, we emulate a cluster of computers within a domain with multiple NICs on each host, that are in turn connected to different ports of our Brocade router. These hosts have Intel Xeon E5620 CPUs and 2 Myricom 10 Gbps Dual NICs. The two extra servers running OSCARS-TE and Floodlight have Intel i7-3770 CPUs. There is no significant load on these servers, as they are used exclusively to run the applications specified. All the OpenFlow control messages use a separate control network running at 100Mbps, that has no significant load.

Our evaluation seeks to address four questions:

1. Understand how interaction of big flows with small flows sharing a link impacts overall network throughput.
2. Quantify the delays involved in traffic offloading for a multi-layered environment, with focus on the delays induced by the creation and deletion of cross connects.
3. Measure the performance degradation in network reconfigurations without using planned updates.
4. Quantify delays involved in traffic optimizations and detection of big flows in our prototype.

1. Small vs. big flows through a circuit

In this experiment, we analyze the impact of currently available TCP congestion control on maximum network throughput. This topic has extensive presence in the literature. However, it is not realistic to assume that all hosts in the network use one particular version of TCP, as network operators often do not have control over the network software stack used by computers. Therefore, we limit our experiments to congestion control algorithms available in our servers. These are Reno, Cubic, Hamilton TCP (htcp) and Highspeed TCP.

We start this experiment creating a 10Gbps circuit to carry traffic between the two domains in Figure 2. In a traditional environment, once the circuit becomes active the SDN controller (or routers at the ends of the circuit) will establish the (logical) link between two switches **A** and **C**. We measured the maximum achievable throughput through the 10G circuit to be ~9.8 Gbps. We then generate traffic patterns intended to replicate the traffic in a network that carries large scientific data transfers and multiple small flows sharing a circuit.

We use the microbenchmark *iperf* to generate competing flows. One of the flows transmits a continuous amount of TCP traffic and have unbounded demands for the duration of the experiment. This flow represents a big data science workload, which involves the movement of a gigabyte-sized file transfer between two domains [11]. The other flows are a combination of small-sized data transfers, created by spawning multiple *iperf* instances, which send random data through the same circuit to the other domain. These small transfers are of *S* bytes each, with *S* uniformly distributed between 512 KB and 4 MB. These sizes were chosen to match the total download sizes of common websites, with some examples listed in Table 1. The benchmark starts C={4,8,16,32} new iperf transfers of size S when the number of active transfers falls below C. Each run lasts for 30 seconds. Because these transfers are bounded by computation (which involves creating iperf processes, allocating buffers in the operating system, establishing TCP connection, etc.) the bandwidth consumed by these transfers are small, usually of a few Mbps. Figure 4 shows the bandwidth consumption of small transfers when C=4 and C=8.

TABLE I. NETWORK CONSUMPTION FOR COMMON WEBSITES

Website Name	*Front Page Html*	*Media Content*
cnn.com	168KB	2.3MB
es.net	36K	1.6MB
en.wikipedia.org	96KB	492KB
youtube.com	400KB	2.1MB
facebook.com	688K	4.3MB

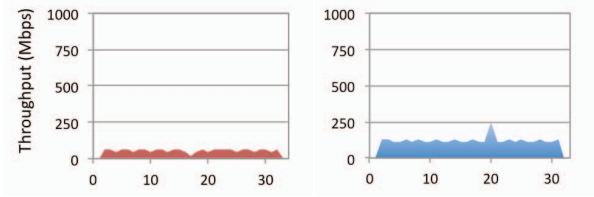

Figure 4: Throughput over time (in seconds) used by small flows when C=4 (left) and C=8 (right). Note: Throughput in Mbps.

Figure 5 shows the throughput achieved using this workload when we use 4 variants of TCP: Reno, Cubic, Hamilton and Highspeed. As we can see, fluctuations on network demand prevents TCP flows from quickly adjusting their bandwidth rates, and result in low link utilization. This happens because TCP congestion control is constantly trying to adjust its throughput, and the attempt to prevent creating more congestion leaves some unutilized bandwidth. The problem gets worse facing higher network delays. Because our testbed is confined to a few data center racks of distance, the RTT be-

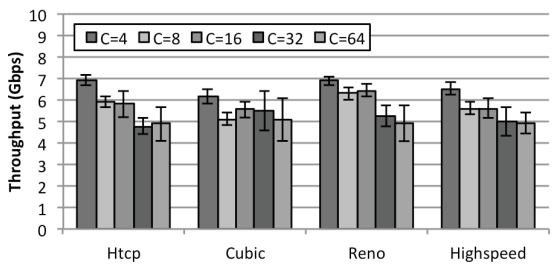

Figure 5: Throughput of a 10G link with a big TCP flow competing with C={4,8,16,…} small concurrent TCP flows.

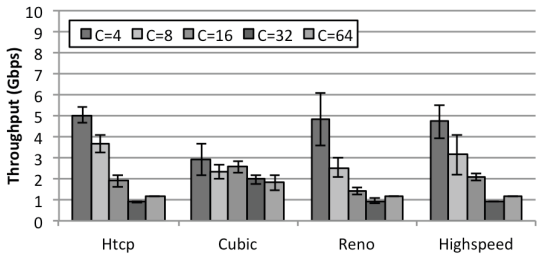

Figure 8: Throughput of a 10G link with a big TCP flow competing with C small TCP flows under 10ms latency

tween nodes is less than 1ms. We simulate slightly higher RTTs by using the Linux Queuing Discipline (qdisc) netem, that delays packet transmission at hosts. We set the delay for 10ms and repeat the same experiment. The results are shown in Figure 6. Note that the throughput achieved using a single flow with higher throughput is also ~9.8Gbps. The reduced throughput facing higher latency comes from the slower bandwidth recovery rate of TCP.

As mentioned earlier, one strategy to mitigate this problem is to use multiple circuits, and split flows based on packet headers, like in ECMP. However, this does not guarantee that one of the links will experience a similar mix of big and small flows, what could still prevent higher levels of utilization on one of the paths. Other approaches would be to reshuffle the flows to different paths as load imbalance starts to happen as in MPLS-TE. However, this approach could lead to non-optimal allocation strategies [3][8], and still lead to the same problem as in ECMP. Instead of such approaches, OSCARS-TE explores allocating extra bandwidth on demand to flows that suffer throughput loss at lower layers.

2. *Network Reconfiguration Overheads*

Before exploring the benefits of OSCARS-TE, we first evaluate the performance of the underlying hardware for dynamic traffic management. This is important to understand if bandwidth relocation across layers can be done fast enough to absorb changes in traffic demands.

Here we focus on the impact of network changes perceived by applications, as application performance is directly related to the performance of a dynamic network. The software used in the network stack is usually sensitive to changes in network availability, and will try to adapt its resource consumption as soon as it identifies problems (video streaming applications on top of UDP for example usually adjust their encoding quality according to resource availability).

We start by measuring the reconfiguration delays to setup XCONs in the transport domain using the OTS. In this case we set up a path that interconnects Cluster 1 to Cluster 2 passing through devices A ↔ X ↔ Y ↔ B ↔ Y ↔ Z ↔ C as shown in Figure 2, and perform a router bypass in node Y, with resulting path A ↔ X ↔ Y ↔ Z ↔ C. Throughout the duration of the experiment, we send small packets with 1 millisecond inter-packet delay. A monotonically increasing sequence number is added to the packets, and the server at cluster 2 inspects them during the reconfiguration, inferring how many packets were lost. Note that this experiment gives us the total time to reconfigure one node from the optical transport

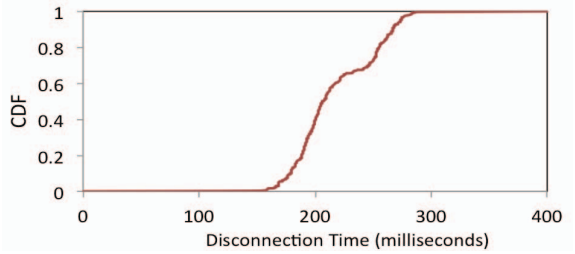

Figure 9: End-to-end view of PXC reconfiguration delay.

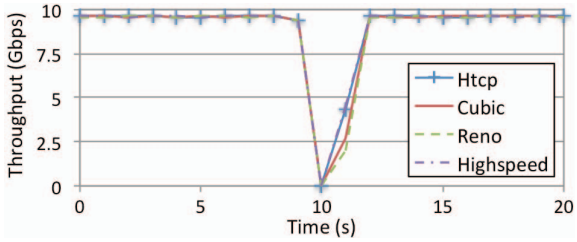

Figure 7: Throughput during one-shot topology update at 10s

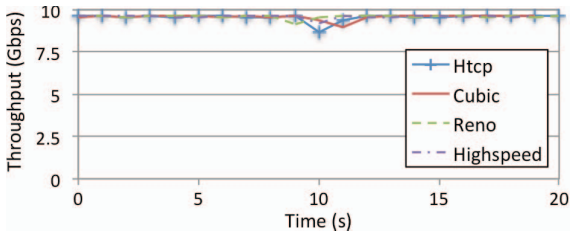

Figure 6: Throughput with planned topology update at 10s

layer, including the software overheads involved in communication with the SDN controller, and the physical layer delays in identifying variations in link availability [16].

Figure 7 shows a CDF of the measured down time. The minimum value measured is 150ms, and average 217ms. Note that this involves not only cross establishment at node Y, but all the control loop between our SDN controller for tearing down two cross connects, setting up a new one, and waiting Ethernet devices to identify the connection [16]. This value can be used in the offloading decision. Given a predicted flow size and expected switching time, we can calculate what would be the gain in offloading a flow to a higher bandwidth path or keeping it in its current path. However, success in this strategy would depend on how good the prediction of flow size is, because if the predictions are not accurate, a flow could be over before a new path is made available.

3. *Regular vs Planned Topology Update*

In this experiment, we show the performance impact of performing topology updates without a multi-layer update plan. We use the path A ↔ X ↔ Y ↔ B ↔ Y ↔ Z ↔ C as a starting point and offload its traffic to path A ↔ X ↔ Z ↔ C. In this case, iperf is used to create a single flow between domains. When T=10s, we request a topology update with manual input, and analyze the throughput over time.

Figures 8 and 9 shows results for regular, one-shot topology update and planned topology update, respectively, for

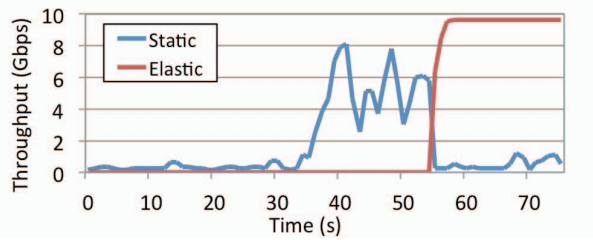

Figure 10: Elastic bandwidth allocation for large transfers. Throughput is limited by contention, and extra resources allow better application-level network performance.

different variants of TCP. In the regular update, offload requests are made by the controller to all the nodes in the path at the same time. We can see that, even though the disconnection time was measured as a few milliseconds, TCP reacts to abrupt changes in link availability more aggressively, and needs a few seconds to recover its maximum throughput. When we use planned topology updates, we do not see this problem, as the traffic is seamlessly moved between paths.

4. *Reacting to demand changes*

In this experiment we evaluate the efficiency of OSCARS-TE in identifying and reconfiguring the network based on the measured demands. We use the same workload as before, with a mix of C concurrent small flows and a big transfer flowing between cluster A and B. However, we start the big flow a few seconds after the beginning of the experiment, to emulate a sporadic big data transfer between the two domains sharing the resources with regular network traffic. The initial path reserved in this experiment is A ↔ X ↔ Y ↔ B ↔ Y ↔ Z ↔ C. The number of concurrent transfers is set to C=16. We also set OSCARS-TE measurement interval Ts to 6s.

Figure 10 shows the measured throughput over time both of the initial static 10Gbps path and the throughput on the elastic path. Initially, from T=0s, only regular traffic is present on the path and throughput oscillates between ~200mbps to 1Gbps. At T~34 the large data transfer is started, which competes for the 10Gbps of the static path with the small transfers, preventing maximum link utilization. OSCARS-TE detects these events after a few seconds, and splits the network traffic responsible for the load imbalance growth. Once the new elastic path is allocated, the big flow is offloaded to it at T~55s, with short flows left in the statically allocated path.

These results show that the initial implementation of OSCARS-TE provides multi-layer orchestration without significant performance degradation. We leave a more extensive analysis of the system and its parameters for future work.

V. CONCLUSION

There are multiple caveats in desigining a multi-layer WAN network management system. In this paper we describe the strategies used by OSCARS-TE, a management system implemented, deployed and evaluated in a multi-vendor multi-layered network. The system uses a SDN controller to intermediate the communication with network devices using OpenFlow (OF). We discuss some of the limitations of the OF protocol to support multi-layer network management and present the solutions we adopted to overcome them.

We show that inefficiencies in inter-data center WAN links can be caused due to TCP congestion control and show how OSCARS-TE can be used to solve this problem. To the best of our knowledge, this is the first practically deployable multi-layer SDN application that addresses multiple practical problems to make cross-layer network orchestration a reality.

ACKNOWLEDGEMENTS

This work was partially supported by the CIAN NSF ERC under grant #EEC-812072 and EC FP7 grant 285939 (ACROSS). We would like to thank Andrew Lake, and Sivaram Balakrishnan for their help and support.

REFERENCES

[1] Cisco Visual Networking Index: Forecast and Methodology, 2012–2017

[2] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., ... & Vahdat, A. "B4: Experience with a globally-deployed software defined WAN". In *ACM SIGCOMM* (pp. 3-14). ACM. 2013.

[3] Hong, C. Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., & Wattenhofer, R. "Achieving high utilization with software-driven WAN". In *ACM SIGCOMM* (pp. 15-26). ACM. 2013.

[4] Doverspike, R. D., & Yates, J. "Optical network management and control". In Proceedings of the IEEE, *100*(5), 1092-1104, 2012

[5] Das, S., Parulkar, G., McKeown, N., Singh, P., Getachew, D., & Ong, L. "Packet and circuit network convergence with OpenFlow" In OSA *Optical Fiber Communication Conference* (p. OTuG1). 2010.

[6] Channegowda, M., Kostecki, P., Efstathiou, N., Azodolmolky, S., Nejabati, R., Kaczmarek, P., Simeonidou, D. "Experimental evaluation of extended OpenFlow deployment for high-performance optical networks" In European Conference and Exhibition on Optical Communication (pp. Tu-1). Optical Society of America (OSA), 2012

[7] Liu, L., Zhang, D., Tsuritani, T., Vilalta, R., Casellas, R., Hong, L., ... & Muñoz, R. "First field trial of an OpenFlow-based unified control plane for multi-layer multi-granularity optical networks". In OSA Optical Fiber Communication Conference (pp. PDP5D-2). 2012.

[8] Pathak, A., Zhang, M., Hu, Y. C., Mahajan, R., & Maltz, D. "Latency inflation with MPLS-based traffic engineering" *ACM SIGCOMM* conference on Internet Measurement Conference (pp. 463-472), 2011.

[9] Sadasivarao, A., Syed, S., Pan, P., Liou, C., Lake, A., Guok, C., Monga, I. "Open transport switch: a software defined networking architecture for transport networks". In ACM HotSDN (pp. 115-120), 2013.

[10] https://github.com/OFWorkshop/OFW-Trema/tree/master/TableProbe

[11] Monga, I., Guok, C., Johnston, W. E., & Tierney, B. Hybrid networks: Lessons learned and future challenges based on esnet4 experience. In Communications Magazine*, IEEE*, *49*(5), 114-121. 2011.

[12] Gerstel, O., Jinno, M., Lord, A., & Yoo, S. B. "Elastic optical networking: A new dawn for the optical layer?" In Communications Magazine, *IEEE. 50*(2), s12-s20. 2012.

[13] Farrel A., A Unified Control Plane: Dream or Pipedream? In International Conference on IP+ Optical Network (iPOP), 2010

[14] Das, S., Parulkar, G., & McKeown, N. "Why OpenFlow/SDN can succeed where GMPLS failed". In European Conference and Exhibition on Optical Communication (pp. Tu-1). OSA. 2012.

[15] ITU. G.709: Interfaces for Optical Transport Network, Feb 2012.

[16] Farrington, N., Porter, G., Radhakrishnan, S., Bazzaz, H. H., Subramanya, V., Fainman, Y., ... & Vahdat, A. "Helios: a hybrid electrical/optical switch architecture for modular data centers". *ACM SIGCOMM*, *41*(4), 339-350. 2010.

[17] Liu, L., Tsuritani, T., Morita, I., Guo, H., & Wu, J. "OpenFlow-based wavelength path control in transparent optical networks: a proof-of-concept demonstration". In European Conference and Exposition on Optical Communications (pp. Tu-5). OSA. 2011.

[18] Reitblatt, M., Foster, N., Rexford, J., Schlesinger, C., & Walker, D. "Abstractions for network update". ACM SIGCOMM (323-334) 2012.