Application-Aware Aggregation and Traffic Engineering in a Converged Packet-Circuit Network

Saurav Das, Yiannis Yiakoumis, Guru Parulkar, Nick McKeown
Department of Electrical Engineering, Stanford University, California 94305, USA
sd2, yiannisy, parulkar, nickm@stanford.edu

Preeti Singh, Daniel Getachew, Premal Dinesh Desai
Ciena Corp., 1201 Winterson Road, Linthicum, MD 2109, USA

Abstract: We demonstrate a converged OpenFlow enabled packet-circuit network, where circuit flow properties (guarantee d bandwidth, low latency, low jitter, bandwidth-on-demand, fast recovery) provide differential treatment to dynamically aggregated packet flows for voice, video and web traffic.
OCIS codes: (060.4253) Networks, circuit-switched; (060.4259) Networks, packet-switched

1. Introduction

IP and Transport networks today do not interact, a fact that leads to several adverse consequences for both networks. For the IP network, this lack of interaction means that the Internet core today is completely based on packet switching – in other words a dependence on expensive, power-hungry and fragile backbone IP routers. It guarantees that the Internet core cannot benefit from more scalable circuit *switches*, nor take advantage of dynamic circuit *switching*. For example, dynamic circuits can recover faster from failures, provide bandwidth-on-demand, or guaranteed low-latency, jitter free paths, all of which are very hard to provide in today's packet-only networks. For the Transport network, the lack of interaction means that it has no visibility into IP traffic patterns and application requirements. Without interaction with a higher layer, there is often *no need* to support dynamic services. As a result the Transport network today remains largely static under the provider's manual control where bringing up a new circuit to support a service can take days.

With the goal of greater interaction via true packet-circuit network convergence, we have recently proposed Software Defined Networking (SDN) and OpenFlow as the unifying architecture and control plane for packet and circuit networks [1,2]. Briefly, SDN principles can be summarized as follows (Fig. 1): separation of data and control paths in packet and circuit networks; flow based datapath where flows (not packets) are the fundamental unit of control; a rich API called OpenFlow(OF) into the switch flow tables for programming and controlling both packet and circuit datapaths; a logically-centralized Controller, running a network operating system, which in-turn provides another API for programming networking applications; and lastly, a means to provide network virtualization by slicing the network and isolating the slices, so that experimental slices can run in parallel to production slices, and backward compatibility is maintained with today's networks. SDN principles allow service providers greater control to run cost-optimized and service-optimized converged packet-circuit networks, where they have maximum flexibility in choosing the correct mix of technologies depending on service needs. With common control over packets and circuits, carriers can innovate outside-the-box by designing networking applications *specifically* taking advantage of the strengths of both kinds of switching technologies. We've previously demonstrated common control over packets switches and different kinds of circuit switches [3,4]. In this paper, we report on a demonstration network in our lab (Fig. 2) and a networking application that benefits from packets and dynamic circuits. This work was demonstrated at the GENI Engineering Conference (GEC8) [5] along with other SDN/OpenFlow based demos.

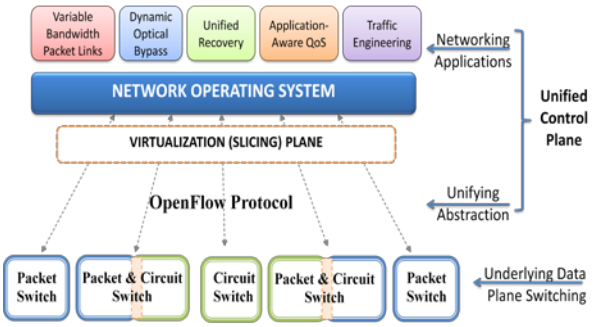

Fig. 1 Unified Architecture of a converged packet-circuit network

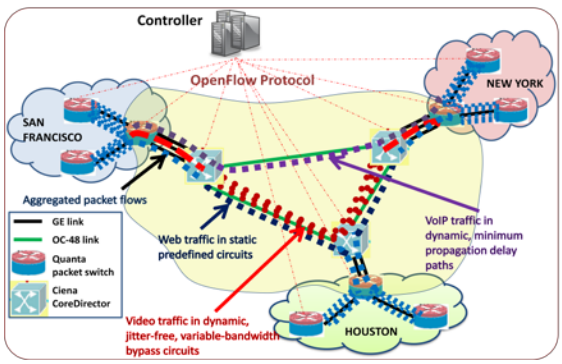

Fig. 2 Demonstration network and applications

2. Networking Capabilities in a Converged Packet-Circuit Network

In this section, we briefly detail new networking applications and capabilities made possible by SDN/OpenFlow based converged packet-circuit networks. It is worth noting that these capabilities could be implemented with IP/MPLS-TE/SNMP working together with ASON/GMPLS/TL-1. But the sheer number of protocols, their distributed nature and their potentially dangerous interactions make the solutions so complex, that no service provider today implements any of the networking applications discussed below.

GMPLS is a superset of the MPLS control plane and retains all the complexity of the latter. It includes a distributed link-state routing protocol like OSPF or IS-IS which has convergence and stability issues if network state changes too fast or too often. This is the fundamental reason why IP networks today do not support dynamic links or dynamic link weights. To extend OSPF and use it in a dynamic circuit network with its effect being felt by another instance of OSPF in the packet network is complete and utter folly. Given the architectural drawbacks and the protocol deficiencies, it is easy to see why GMPLS, originally devised as a Unified Control Plane (UCP), has never been used commercially as one, despite a decade of development and standardization work. GMPLS work has *devolved* from being a UCP idea to one meant *only* as a control plane for Transport networks, and even here it has found limited use, predominantly as vendors' proprietary implementations.

We are more interested in OpenFlow as a multi-layer UCP instead of merely a Transport network control plane. Importantly, with our architectural changes in *both* networks, a*ll* the capabilities outlined below can be developed in the Controller, requiring *no other protocol* but the OpenFlow protocol for operation.

i) Dynamic Packet Links: An SDN/OpenFlow network eliminates the need for distributed routing protocols within the Controller's domain, as the switches do not make routing decisions. With centralized-decision making, it enables convergence-free dynamic packet-link creation, which is non-disruptive to packet flows elsewhere in the network. Dynamic packet-links (supported by dynamic-circuits) allow the service provider more flexibility in operating their networks, without requiring a full mesh of packet-links.

ii) Dynamic Service-aware Aggregation and Mapping: Aggregation is necessary in WANs so that core-routers have more manageable number of rules in their forwarding tables (a few hundred thousand instead of millions). Such aggregation can take the form of IP supernetting (or CIDR eg. using /20s instead of individual /24s) or encapsulation/insertion of labels and tags. Often such aggregation is manually configured, static and error-prone. But OpenFlow allows great flexibility in defining flow granularity, with which we can dynamically and programmatically perform aggregation, simply by changing the definition of a flow. For example, if all flows from a customer take the same route in the core, we can perform aggregation by entering just a single flow-table entry to match on the customer's source-IP address. Note that the packets themselves do not change, just their representation in the packet-switches flow table changes. And so, in the core we can now collectively reference all flows with a single aggregated flow bundle. We could also perform supernetting or label insertion if desired. And we can go further by differentiating between traffic types from the same customer, by creating separate bundles (or aggregates) that match on the customer's source IP address *and* the tcp/udp port in the packet header. Finally we can map these application/service specific bundles to different circuits, in order to give differential service-specific treatment to the bundles in the circuit network.

iii) Application-aware Routing: By dynamically creating circuits for application/service specific bundles, we can tailor the circuit to have characteristics beneficial for the application or service (eg. the path over which the bundle is routed). For instance, VoIP traffic can benefit from low latency paths. For such bundles, we could dynamically create a circuit between source-and-destination packet switches, where the circuit path is the one with the smallest propagation-delay. Another example is video traffic – for video, low-latency is not as important as low-jitter. Again we can dynamically create a video-circuit and route it over the non-shortest-propagation path in the circuit topology. But importantly, such a path still bypasses potential intermediate packet-switches between source and destination, and thus avoids potential switching delay-variations (jitter) in those switches.

iv) Variable Bandwidth Packet Links: We can selectively monitor the bandwidth usage of the circuits that make up a packet-link, and as usage varies, we can dynamically vary the bandwidth allocated to those circuits. For example, voice and http traffic may not be as bandwidth-hungry as video traffic. By selectively monitoring the bandwidth consumption of a video-circuit, we can dynamically change its size when sustained traffic surge is observed, thereby relieving congestion in the packet-link.

v) Unified Recovery: Finally, with global knowledge and centralized decision making, we can recover from network failures selectively keeping application/service needs in mind. For example, video traffic could be circuit-protected with pre-provisioned bandwidth, while voice could be dynamically re-routed in the circuit topology and http traffic bundles could be re-routed in the packet topology. Unified Recovery allows for multiple packet and circuit flow recovery mechanisms to co-exist without inefficient duplication of resources for fault-tolerance in multiple layers.

3. Application-aware Aggregation & Traffic Engineering Demonstration

The demonstration network consists of seven GE Quanta LB4G packet switches and three Ciena CoreDirector (CD) hybrid packet-circuit switches. The CDs have Ethernet interfaces connected to the Quantas, and SONET/SDH interfaces connected to each other as shown in the topology in Fig. 2. The demo-network *emulates* a wide-area network between 3 cities (SF, Houston, NY) with edge/core packet switches connected over the wide-area by Transport network elements such as the CDs. All the switches are OpenFlow enabled i.e. they have an OpenFlow client that runs in the switch and communicates with an external Controller using the OpenFlow protocol over an out-of-band Ethernet network. The Controller is a PC that runs a network operating system (NOX [6]) on which we build our networking applications. The network also includes 6 PCs (not shown) with various kinds of software traffic generators that generate http (tcp port 80), voice (tcp port 5060) and video (udp port 1234) traffic.

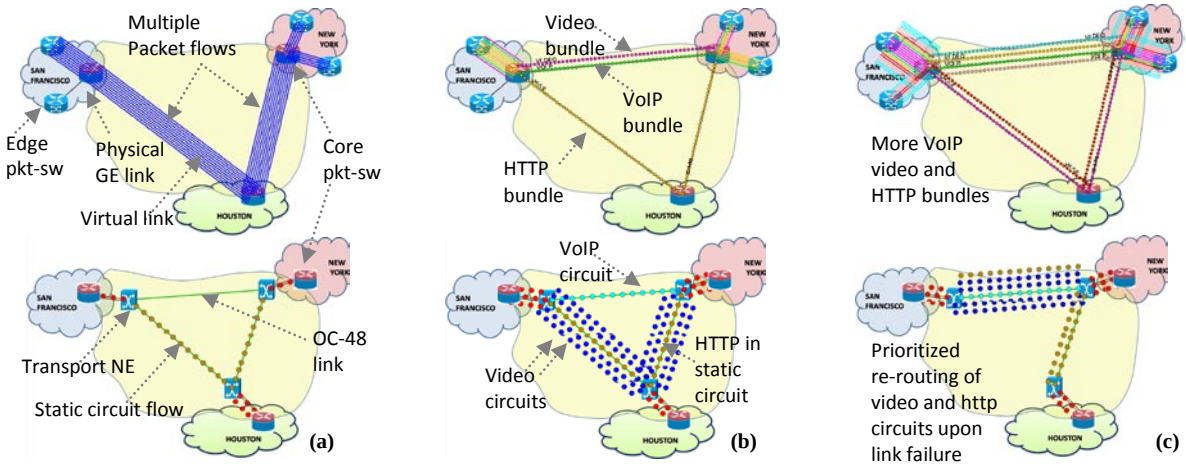

Fig. 3 GUIs showing real-time network state

We have created two GUIs that display real-time network state. The upper GUI in Fig. 3 displays the packet topology. The links shown *within* the cities correspond to physical GE links between the Quantas and also between the Quantas and the CD's Eth ports. The links *between* the cities are virtual packet links supported by static or dynamic circuits created in the circuit network. It also shows packet flows currently routed in the network. The lower GUI in Fig. 3 displays the fiber topology, the CDs and *only* the core packet switches that connect to them. It also shows the circuits that have packet flows mapped into them. For eg, in Fig. 3(a) static 'configured' circuits allows the packet topology to be completed over the wide-area, so that the Controller can then route all traffic between the cities - in this step un-aggregated traffic is routed from SF to NY via the Houston core-packet switch.

In Fig. 3(b), we dynamically aggregate the 3 different traffic types from the same customer into 3 bundles, and route the video and voice bundles over a dynamically created virtual packet link between SF and NY. This link is supported by a voice-circuit of lower bandwidth with a direct-propagation path between SF and NY, and higher bandwidth, non-shortest path video-circuits that bypass the Houston core packet-switch. We also monitor the bandwidth of the video-circuit and vary it to match usage. In Fig. 3(c), we show that we can similarly aggregate more customer-specific and application-specific bundles and multiplex them over the specialized circuits. Finally we show an instance of dynamic re-routing upon link failure in the circuit network. In this case, the Controller re-routes existing static and dynamic circuits on that link, with priority given to the video circuits over the http (static) ones.

4. Summary

We believe the SDN/OpenFlow architecture and control plane can create mutually beneficial interaction between IP and Transport networks by enabling new capabilities at the packet-circuit interface. We've outlined such capabilities and demonstrated a networking application that uses them to provide application-aware aggregation and TE. A video of this demo, more details with reference code, and directions to re-create it are available in [7].

5. References

[1] S. Das, G. Parulkar, N. McKeown, "Unifying Packet and Circuit Networks", Below IP Networking (BIPN), November 2009
[2] Packet and Circuit Convergence (PAC.C) with OpenFlow wiki: http://openflowswitch.org/wk/index.php/PAC.C
[3] S. Das, et. al. "Packet and Circuit Convergence with OpenFlow", OFC/NFOEC, March 2010
[4] V. Gudla, et. al. "Experimental Demonstration of OpenFlow Control of Packet and Circuit Switches", OFC/NFOEC, March 2010
[5] GENI Engineering Conference, July 2010: http://groups.geni.net/geni/wiki/GEC8DemoSummary
[6] http://noxrepo.org
[7] http://openflowswitch.org/wk/index.php/Aggregation_on_a_Converged_Packet-Circuit_Network