SOFTWARE DEFINED NETWORKING FOR VIDEO: OVERVIEW & MULTICAST STUDY

Wai-tian Tan, Herb Wildfeuer, John Apostolopoulos
Chief Technology and Architecture Office, Cisco Systems, Inc.

ABSTRACT

Software Defined Networking (SDN) is an architectural trend in networking towards the use of centralized controllers to improve global visibility and simplify various network operation tasks. Due to their need for QoS, sizable traffic involved, and dynamic nature, video applications are particularly suitable candidates for dynamic interaction with SDN. This paper provides an overview of different general ways that video applications have interacted with networks, and outline what opportunities SDN provides. We then develop and evaluate several methods that exploit SDN to construct multicast video trees and show that over realistic topology, SDN optimized trees can support 60-90% more traffic.

Index Terms— SDN, video multicast

1. INTRODUCTION

Current deployed networks typically consist of a large number of nodes operating in a distributed manner for scalability and resiliency. However this leads to various challenges and inefficiencies. Software Defined Networking (SDN) is an important technology trend that promises significant benefits. "SDN" has a variety of interpretations. A common SDN theme is that networks are evolving from a fully distributed system to one with more centralized visibility and control. One feature of SDN that is fairly common to all definitions is separation of some (or all) of the network controller plane functions from the distributed networking elements (e.g. switches, routers, gateways) to a centralized (or less distributed) component, i.e. the *controller*.

A general architecture of SDN is shown in Fig 1, where an SDN controller is shown to be a layer between application servers and network elements. Its function is to serve as a bridge to make configuration, deployment and operation of network elements easier for the benefits of applications. The features of the control plane moved from the network elements to the SDN controller is a design option. Under some implementations/definitions, the entire control plane is implemented in the controller with the forwarding plane of the network controlled via API's call from the controller to the network elements (e.g., OpenFlow [1] API). Other implementations leave select control plane functions in a distributed manner using existing protocols, but implement other control plane functions (e.g., QoS assignment, Call Admission Control, security features) in the centralized controller.

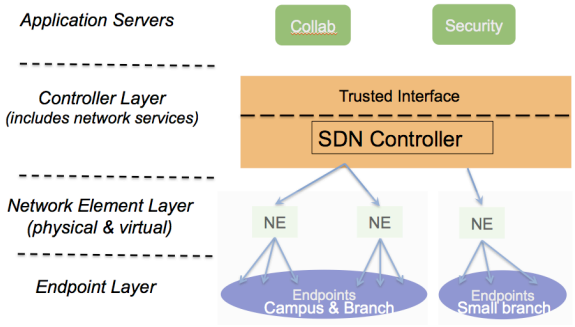

Fig 1. General architecture of SDN

SDN provides simplicity by hiding the complexity of interoperating across diverse sets of network elements by providing easy to use "North-bound" interfaces (i.e., to entities north of controller in Fig 1), and can be used to simplify both static and dynamic configuration of networks. Dynamic configuration using SDN is particularly relevant for video applications, both due to higher sensitivity to network conditions, and the long-lived and high bandwidth flows warrant more expensive involvements of a centralized controller.

Our goal in this paper is twofold. First, we provide an overview of different general ways that video applications interacted with networks, and outline the opportunities introduced by new interactions with SDN. This is covered in Section 2. Then, in Section 3, we develop and evaluate several methods that exploit SDN capabilities to varying degrees to construct multicast distribution trees. Through simulation using realistic network topology, we show that a network can support 60% to over 90% more traffic with SDN optimized multicast compared with simple shortest paths.

2. APPLICATION & NETWORK INTERACTIONS

The controller interacts with the network elements using "South-bound" API's called from the controller to the individual network elements. Much activity is currently underway to define and model these API's so that SDN can be implemented in heterogeneous networks with a variety of controller plane function split between the controller and the network elements. Common is the use of Yang models with REST API's (e.g., NETCONF) [2].

Regarding services that SDN provides ("North-bound" API), one important concern is how and who can be allowed access. To categorize how SDN services can be utilized by vid-

eo applications, we find it convenient to distinguish along the axis shown in Fig 2. Specifically, we are interested in distinguishing between applications that are "over-the-top" that use network for basic connectivity service without attempting to model and work better with the network versus "network-aware" applications that do, and how SDN brings changes to both of these classes of applications.

2.1. "Over-the-top" App/ Non-SDN Network

These applications expect only basic transport service from the network, and implement additional measures such as security and reliability using end-point mechanisms alone. These can often work well, though usually not preferred for mission critical applications. There can be network services utilized via priority markings in the data packets, but there is limited opportunity for network services of significant value or sophistication.

2.2. Network Aware App/ Non-SDN Network

Even before SDN, video applications have worked closely with the network to ensure the best available quality. This can happen in two fronts. There are explicit protocols for application interaction with non-SDN networks. Examples of this are RSVP for bandwidth reservation as well as admission control, and IGMP for creation of multicast sessions, both of which can be initiated by the application endpoints.

Furthermore, business video applications have long appreciated the importance of understanding and controlling the network to avoid unpleasant experience when video or other traffic overloads a network. Fig 3 shows a current non-SDN solution where a video call manager maintains a static (often manually configured) model of the underlying network for purposes such as call admission control. While this is current practice, it is clear that the model can get gravely out-of-date, e.g., when a link goes down or when there are sizable changes in traffic patterns. With up-to-date network status and topology from SDN, applications can make call decisions based on actual network availability.

2.3. "Over-the-top" App/ SDN Network

While existing discovery mechanisms can be used to announce SDN based network services directly to end-user applications, there is no good security model to grant network control to applications without a trust relationship with the network/SDN. One possibility is for the network operator to install and operate trusted "orchestration applications" to selectively grant SDN access. Hypothetically, if Comcast wants to selectively mark all Netflix traffic high priority, it can utilize various application visibility features in existing equipment to identify Netflix flows to a Comcast orchestration application that has SDN access. This orchestration app will instruct the network via the SDN controller to dynamically mark packets for these flows without requiring changes to Netflix servers or clients. A prototype utilizing deep packet inspection with SDN to prioritize YouTube is given in [3].

	Non SDN	With SDN
Network-aware applications	2.2	2.4
"Over-the-top" applications	2.1	2.3

Fig 2. A classification of video applications according to whether they attempt integration with the network, and how SDN brings changes. Discussed in the sections noted.

2.4. Network Aware App/ SDN Network

SDN implications to network-aware applications are significant. Rather than indirect, complex, and less accurate ways to gain visibility and control of the network, such as using bandwidth model of Fig 3, those can now be accessed directly by the applications. Dynamic configuration of packet priority for video flows using SDN has been widely demonstrated in industrial forums. For business applications requiring other advanced service from the network, SDN reduces complexity by providing direct API calls.

As we will explore further in Section 3, SDN provides not only additional useful information on true network topology and link usage statistics, but also interfaces to flexibly configure optimized multicast tree for video distribution.

3. MULTICAST VIDEO USE CASE

The significance of SDN architecture/paradigm for video based applications can best be described through a high value use case. Here we describe the application/network interaction for a Network Aware Application/SDN based network for efficient distribution of multicast video.

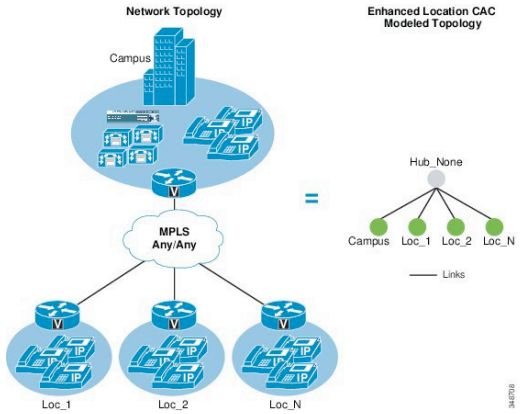

Fig 3. Before SDN, business video applications build static network models to support network features such as call admission control.

3.1. Problem Description and Conventional Approach

In this use case, the non-SDN approach is to use layer-3 multicast protocols with applications interacting with the network using the IGMP protocol. The non-SDN network would create multicast trees in a fully distributed manner using Protocol-Independent Multicasting (PIM) technology, resulting in a multicast tree that is a union of the respective shortest paths from the source to each destination. As will be discussed later, this will be designated as the *Default* scheme.

SDN based networks can implement efficient video distribution in the network both using traditional layer-3 protocols and network overlays. The use of efficient overlay distribution trees in the network is possible since the control layer implementation is no longer limited to the use of distributed protocols. In layer-3 based multicast, the endpoints are capable of IP-multicast generation/consumption, with application (server) interaction via north-bound API calls to the controller and tree creation at the controller utilizing a global topology and load awareness of the network. We demonstrate how this centralized control plane can achieve significant improvements over the fully distributed non-SDN approach.

3.2. Multicast Tree Construction

The general problem of constructing efficient multicast trees is well studied. For a single source multicast, constructing the minimum hop-count tree is the well-known NP-hard minimum Steiner tree problem that nevertheless has approximate solutions [4]. When network coding is allowed, explicit algorithms are known to achieve capacity [5]. When there is more than one multicast tree, the capacity region can still be characterized [6] but there is no known realization to achieve it. Multicast can also be performed by end-points only without infrastructure support, as outlined in [7]. It is shown that higher efficiency is possible by using multiple trees, each for a "split-stream" of a video [8]. Furthermore, [9] shows that in peer-to-peer multicast where bottlenecks are in uplink bandwidth routing can indeed be optimal. Hybrid schemes involving both end-host and infrastructure are also possible [10].

While efficient algorithms for contructing optimal "multi-source" multicast trees continue to hide behind an NP-hard hurdle, it is arguable whether the optimal solution that may require re-routing of established traffic is practical after all. In [11], the authors established a practical routing scheme for NP-hard multi-commodity flow problem that does not require re-routing existing flows, and employed the insight that minimizing "interference" with future flows should be considered when making routing decisions.

In this Section, we explore several multicast tree construction methods that exploit topology and link usage statistics to varying degrees, including a version that attempts to limit future "interference." Specifically we distinguish between the following multicast tree construction methods:

- **Shortest Path (Default):** is the union of the shortest paths between the source and each multicast destination. It achieves minimum delay without regard to link cost.
- **Minimum Bandwidth (MinBw):** true minimization of link cost is NP-hard. Instead we use topology information only to build a low link-cost tree by sequentially adding the lowest "cost" path of any remaining multicast destination to any node in the current tree. Hop count is used as cost. The above two methods do not require link usage.
- **Shortest Available Path (SAP):** exploits dynamic link usage to find the union of shortest paths with enough bandwidth between the source and each destination. Required computation is low, as computation of routes is necessary only when a link transitions to have no bandwidth.
- **Minimum Available Bandwidth (MinABw):** is similar to MinBw but exploits dynamic link usage statistics and uses hop count of path with available bandwidth as "cost".
- **Balance:** is a version of MinBw that avoids links with low available capacity by using the sum of 1/link-capacity along a path as "cost."

All methods above do not re-route existing traffic, and is suitable for use in normal operation. As noted earlier, SDN can provide up-to-date information about link outage, and unexpectedly high network load, e.g., caused by flash news.

	Require link usage	Recompute Route
Default	N	N
MinBw	N	Y
SAP	Y	seldom
MinABw	Y	Y
Balance	Y	Y

Table 1. Summary of characteristics of various multicast tree construction methods.

3.3. Simulation Results

To illustrate the potential benefits an SDN can bring to multicast video applications, we perform simulation on a network shown in Fig.4 that represents the network topology of an ISP (Cogent) in 2010. The network is captured by Topology Zoo project [12] and has 197 nodes and 245 edges.

Fig 4. The network of ISP Cogent as captured by Topology-Zoo in 2010 has 197 nodes and 245 edges

Link bw	5	10	20	50	75	100
Default	9.3	20.1	42.8	113.3	171.9	230.9
MinBw	10.3 (+10.3%)	22.0 (+9.5%)	47.6 (+11.1%)	130.4 (+15.0%)	197.0 (+14.6%)	265.8 (+15.2%)
SAP	14.5 (+55.8%)	32.2 (+59.9%)	68.9 (+60.7%)	182.1 (+60.7%)	277.2 (+61.1%)	372.8 (+61.6%)
MinABw	16.6 (+78.1%)	36.4 (+80.8%)	78.7 (+83.5%)	206.0 (+81.8%)	312.1 (+81.5%)	419.4 (+81.8%)
Balance	18.4 (+97.4%)	39.2 (+94.8%)	83.1 (+93.9%)	215.7 (+90.3%)	327.5 (+90.4%)	439.4 (+90.4%)

Table 2: Number of supported flows of various schemes using different link bandwidth. Percentage improvement over "default" is shown in bracket.

We assume each link in the network is full duplex of identical capacity of B units in each direction. Multicast traffic is generated, with each session having a source and a number of destinations that are randomly chosen. The number of destinations is uniformly chosen between 2 to 10. We vary the link bandwidth B in the set {5, 10, 20, 50, 75, 100} to simulate different levels of multiplexing. For each of the multicast construction method in Table 1, we progressively add multicast sessions until the network cannot support more. The experiment is repeated 100 times and the average number of supported sessions is reported in Table 2.

We see that the schemes *Default* and *MinBw* that do not take into account run-time link usage perform significantly worse. Specifically, *MinBw* achieves only about 10% improvement over *Default* despite significant computation overhead. However, *MinBw* does achieve the lowest tree-size of 27.3 links on average, as shown in Table 2, compared to 31.6 of *Default*. In contrast, the schemes *SAP*, *MinABw* and *Balance* that exploit run-time link usage perform significantly better, with 60-90% improvement over *Default*, with modest tree-size of 33.0, 29.9, and 32.1, respectively. Of the three, SAP perform worst in terms of number of supported flow and tree-size, but can be implemented with much lower complexity. Intuitively, since smaller tree-size favors packing more multicast session in the same network, one would expect *MinABw* to perform better than *Balance*. The fact that *Balance* outperforms *MinABw* by about 5% is indication that balancing of load is a useful objective during tree construction, especially when the two schemes require essentially the same amount of computation.

	Average Tree Size
Default	31.6
MinBw	27.3
SAP	33
MinABw	29.9
Balance	32.1

Table 3: The average tree size (in links) for various multicast tree construction schemes.

Fig.5 shows the result in Table 1 again, with y-axis showing the number of supported sections normalized (i.e., divided) by the link-bandwidth B. As we can see, the schemes that do not use run-time link usage, shown dotted, perform significantly worse over the entire range of link-bandwidth B. Also, simple combining of link usage, as implemented in *SAP* can realize significant improvements, while additional improvement of about 20% can be further achieved by *Balance*. True achievable upper bounds for multicast routing networks are not known, even though the capacity region has been characterized when network coding is allowed [4].

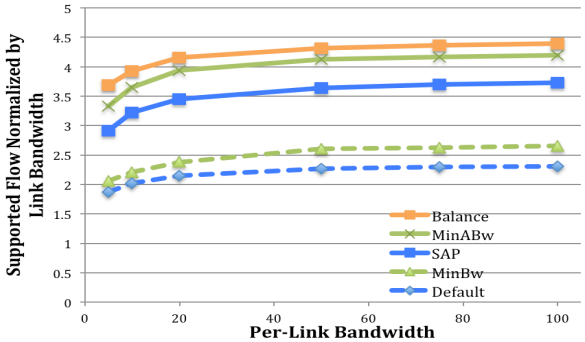

Fig 5. Number of supported flows for different schemes normalized by link-bandwidth.

4. SUMMARY AND DISCUSSION

In this paper, we provide an overview of different general ways video applications have interacted with the network, and outline the difference with the advent of SDN. As an illustration of the benefits of SDN, we demonstrate through simulation over realistic network topology that SDN-optimized video tree construction can support 60-90% increases in traffic load. Furthermore, these SDN-optimizations are practically deployable.

In our multicast tree constructions, we have limited ourselves to the more readily available information in SDN such as topology and available link bandwidth. At the cost of added complexity, path delay and losses can also be incorporated. SDN can also greatly simplify and optimize bandwidth sharing by acting as a central bandwidth broker. Rather than relying on distributed multicast congestion control algorithms that do not communicate across sessions and are reactive, SDN can determine and communicate sending rates in both an informed and fast manner -- before any packets are sent.

REFERENCES

[1] N. McKeown, *et al*., "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, 2008.

[2] M. Bjorklund, "YANG – A Data Modeling Language for the Network Configuration Protocol (NETCONF)," in *IEFT RFC 6020*, October 2010.

[3] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, P. Tran-Gia, "SDN-based Application-aware Networking on the Example of YouTube Video Streaming," in *Proceedings of Second European Workshop on Software Defined Networks*, Berlin, Germany, October, 2013.

[4] M. Charikar, *et al*., "Approximation Algorithms for Directed Steiner Problems," in *Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms*, pp 192-200, January 1998, San Francisco, CA.

[5] S. Jaggi, *et al*., "*Polynomial Time Algorithm for Multicast Network Code Construction*," in IEEE Trans. Info. Theory, Vol. 51, Np 6, pp 1973-1982, June 2005.

[6] X. Yan, R.W. Yeung and Z. Zhang, "The Capacity Region for Multi-source Multi-sink Network Coding," in *Proceeding 2007 IEEE International Symposium on Information Theory*, Nice, France, June 2007.

[7] Y.H. Chu, S.G. Rao, S. Seshan and H. Zhang, "A Case for End System Multicast," *IEEE Journal in Selected Areas in Communications*, Vol. 20, No 8, pp 1456-1471, October 2002.

[8] M. Castro, *et al.*, "Splitstream: High-bandwidth Multicast in Cooperative Environments," in *Proceedings of 19th ACM Sym. on Operating Systems Principles (SOSP),* Bolton Landing, New York, October 2013.

[9] S. Sengupta, M. Chen, P.A. Chou and J. Li, "On Optimality of Routing for Multi-source Multicast Communication Scenarios with Node Uplink Constraints," in *Proceeding of International Sym. Info Theory*, Toronto, Canada, July, 2008.

[10] V.N. Padmanabhan, H.J. Wang, P.A. Chou and K. Sripanidkulchai, "Distributing Streaming Media Content using Cooperative Networking," in *Proceedings ACM NOSSDAV*, Miami Beach, Florida, May 2002.

[11] K. Kar, M. Kodialam and T.V. Lakshman, "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications," *IEEE Journal in Selected Areas in Communications*, Vol. 18, No 12, pp. 2566-2579, December 2000.

[12] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden and M. Roughan, "The Internet Topology Zoo," *IEEE Journal in Selected Areas in Communications*, Vol. 29, No 9, pp. 1765-1775, October 2011.