A Scalable Source Multipath Routing Architecture for Datacenter Networks

Wen-Kang Jia
Smart Network System Institute
Institute for Information Industry
Taipei, Taiwan
wkchia@iii.org.tw

Gen-Hen Liu
Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
gordon.cm99g@nctu.edu.tw

Yaw-Chung Chen
Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
ycchen@cs.nctu.edu.tw

Abstract—**The continuous growth in traffic volumes on modern Data Center Networks (DCNs) can be classified into either delay- or throughput-sensitive applications. In order to providing quality delivery to such applications, it is important that multipath routing and/or forwarding can be supported in the DCNs. However, some inherent problems of multipath forwarding have not been totally solved by existing solutions. Aiming to balance all these requirements for the multipath DCNs, we propose a novel solution—*Code-Oriented eXplicit MultiPath (COXMP)* scheme which evenly distributes traffic along multiple paths, resulting in better network throughput, delay, jitter, and resilience. The simulation results show that COXMP outperforms existing solutions and achieves a reasonable tradeoff between scalability and overhead.**

Keywords—Multipath forwarding; Data center network (DCN); Fault-Tolerant

I. Introduction

To meet various requirements of cloud applications such as massive data processing and large-scale distributed computing, many cloud datacenters have built their own private networks—Data Center Networks (DCNs) [10]-[14]. A DCN commonly comprises tens of thousands of computers interconnected with commodity Ethernet switches. DCNs are very different from the Internet in many respects, such as topologies, addressing, routing, forwarding, and traffic patterns. Because of these differences, DCNs attract considerable attention and become a research hotspot. Unfortunately, today's DCN still cannot meet the increasing demands placed on it. Issues such as throughput, packet loss, delay and jitter are difficult problems not well addressed in the current DCN architecture.

Due to time/space complexity and other cost restrictions on DCNs, the current routing schemes only consider a single shortest path from the source to the destination. However, single-path routing cannot achieve the full utilization of all network resources including routing node resources (e.g., computation time and memory space) and link resources (e.g., bandwidth) available to the DCN. Conversely, Multipath routing is the dispersing of traffic between the communication parties over multiple paths through the DCN. Multipath routing algorithms calculate multiple paths between nodes, and the choice of paths is performed by the sources because optimized usage of the paths is application specific in most cases. It provides each host more available resources within the network and allows much better utilization and performance. It is well understood from the queuing theory that we can achieve much better overall and individual performance such as lower queuing delays, less packet losses, and higher throughput by sharing more of the available network resources [1].

Since most of the modern DCN architectures utilize multiple minimal paths between any two hosts, e.g., a fat-tree topology. Therefore, multipath has become a reasonable choice for DCN as it may allow high-throughput and load-balancing between arbitrary given pair of hosts. In the consideration that a network device or link may encounter occasional failure, multipath will take advantage of fault tolerance (a.k.a. high-availability) in the DCN to find other paths for avoiding traffic congestion.

Nowadays, source routing has become more popular on DCNs [13]. Previous investigation shows that source routing can be achieved by establishing multiple path pairs through source control in separated packets [11], but it is not easy to represent a complete end-to-end multicast and multipath information in an individual packet header by a cost effective encoding method [2], as exemplified by *Line Speed Publish/Subscribe Inter-Networking (LIPSIN)* [6], which supports stateless multicast since forwarding information is inside the packet, thus routers do not need to store multicast state. Nevertheless, due to the probabilistic nature of Bloom filters, LIPSIN is susceptible to false forwarding decisions. It has been repeatedly reported that when the false positive probability (FPP) of a Bloom filter exceeds a small value, there is a sharp decline in bandwidth efficiency and most of the bandwidth resources are spent in transmitting false-positives. In other words, the overhead may grow and lead to the performance degradation since the proportion of overhead in multimedia is relatively heavy. Hence, the technique faces severe scalability issues with respect to the size of the multicast group where many links need to be encoded. In this paper, we propose a new mechanism for achieving the goal with less network resources.

II. The Proposed Scheme

We had proposed a novel multicast routing architecture COXcast previously in [4], to deal with various issues regarding DCNs today. COXcast was designed for applications that utilize multicast routing for communication with multiple receivers. The goals of COXcast are to improve the scalability, performance efficiency of the current multicast routing

978-3-901882-77-7 (c) 2015 IFIP

protocols, and provide higher flexibility in the deployment of new multicast services. Its adequate scenario is in a great quantity of medium-scale groups with few thousands of participants. COXcast introduces three elements 1) source-specific *Multicast Channel-specific IDentifier (MCID)*; 2) node-specific KEYs, which are stored at each intermediate switch/router; 3) node-specific *Output Port Bitmaps (OPBs)* at each intermediate switch/router. The OPB which is derived from MCID$\equiv$OPB$_1$ (mod KEY$_1$) $\equiv$OPB$_2$ (mod KEY$_2$)$\equiv$… $\equiv$OPB$_n$ (mod KEY$_n$) can represent the overall multicast delivery tree in a DCN. The proof and further details can be found in [5].

In this section, we present a multipath self-routing scheme–COXMP for applications in datacenter networks (DCNs) based on a variant of COXcast. In COXMP, the considered DCNs are characterized entirely by COXMP-enabled switches, and the scheme can be adopted to *Software Defined Networking (SDN)* framework [9] in the future. Our goals are to improve the scalability and efficiency performance of the current multipath routing protocols, and provide higher flexibility in the deployment of new multipath services in large-scale DCNs.

COXMP is adapted from the basic COXcast in a similar way: which cleverly replaced the "select-all OPBs" with "select-one OPBs", and played it with different effects. In practice, the COX-enabled routers and switches often have double duties, so a packet with COX header could also be used for multicasting, or for multipath routing.

In order to support the COXMP, source node should maintain two arrays: the KEY's set and OPB's set through the multiple optimal (shortest) paths in the network are collected by inter-operating with each intermediate switch along each path, using appropriate protocols. For example, in the case of reference DCN in Figure 1, when a source node *s* wants to establish a connection with destination node *d*, we first enumerated 8 end-to-end *Equal-Cost Multipaths (ECMPs)* from *s* to *d* as follows:

$$\mathbf{ECMP}(s,d)=\left\{\begin{array}{l} s\to E1\to A1\to C1\to A4\to E4\to d\\ s\to E1\to A1\to C2\to A4\to E4\to d\\ s\to E1\to A1\to C1\to A3\to E4\to d\\ s\to E1\to A1\to C2\to A3\to E4\to d\\ s\to E1\to A2\to C1\to A4\to E4\to d\\ s\to E1\to A2\to C2\to A4\to E4\to d\\ s\to E1\to A2\to C1\to A3\to E4\to d\\ s\to E1\to A2\to C2\to A3\to E4\to d \end{array}\right\} \qquad (1)$$

Considering an end-to-end multipath in the reference network comprising a set of disjoint nodes

$$E=\{E1, A1, A2, C1, C2, A3, A4, E4\} \qquad (2)$$

with $n=8$ elements.

For any given disjoint node there exists a unique key k_i. Suppose k_i are positive integers which are pair-wise co-prime with each other, we then have the disjoint forwarding set

$$\left\{\begin{array}{l} k_{E1}\xrightarrow{p_{E1}=1}k_{A1}\xrightarrow{p_{A1}=1}k_{C1}\xrightarrow{p_{C1}=4}k_{A4}\xrightarrow{p_{E4}=4}k_{E4}\xrightarrow{p_{E4}=4}\\ k_{E1}\xrightarrow{p_{E1}=1}k_{A1}\xrightarrow{p_{A1}=2}k_{C2}\xrightarrow{p_{C2}=4}k_{A4}\xrightarrow{p_{E4}=4}k_{E4}\xrightarrow{p_{E4}=4}\\ k_{E1}\xrightarrow{p_{E1}=1}k_{A1}\xrightarrow{p_{A1}=1}k_{C1}\xrightarrow{p_{C1}=3}k_{A3}\xrightarrow{p_{A3}=4}k_{E4}\xrightarrow{p_{E4}=4}\\ k_{E1}\xrightarrow{p_{E1}=1}k_{A1}\xrightarrow{p_{A1}=2}k_{C2}\xrightarrow{p_{C2}=3}k_{A3}\xrightarrow{p_{A3}=4}k_{E4}\xrightarrow{p_{E4}=4}\\ k_{E1}\xrightarrow{p_{E1}=2}k_{A2}\xrightarrow{p_{A2}=1}k_{C1}\xrightarrow{p_{C1}=4}k_{A4}\xrightarrow{p_{E4}=4}k_{E4}\xrightarrow{p_{E4}=4}\\ k_{E1}\xrightarrow{p_{E1}=2}k_{A2}\xrightarrow{p_{A2}=2}k_{C2}\xrightarrow{p_{C2}=4}k_{A4}\xrightarrow{p_{E4}=4}k_{E4}\xrightarrow{p_{E4}=4}\\ k_{E1}\xrightarrow{p_{E1}=2}k_{A2}\xrightarrow{p_{A2}=1}k_{C1}\xrightarrow{p_{C1}=3}k_{A3}\xrightarrow{p_{A3}=4}k_{E4}\xrightarrow{p_{E4}=4}\\ k_{E1}\xrightarrow{p_{E1}=2}k_{A2}\xrightarrow{p_{A2}=2}k_{C2}\xrightarrow{p2=3}k_{A3}\xrightarrow{p_{A3}=4}k_{E4}\xrightarrow{p_{E4}=4} \end{array}\right\} \qquad (3)$$

For each next-hop edges of given disjoint node, we can merge multiple Output Port Indexes (OPIs) for given different ECMPs into a single OPB with simple union. Then, we have two arrays contain all the node-specific keys $\bar{k}=\{k_1,k_2,...,k_n\}$ of set *E* and corresponding OPBs $\bar{b}=\{b_1,b_2,...,b_n\}$. Note that for any given OPB there exists an upper boundary such that $b_i<k_i$.

In the previous example, based on the proposed scheme, the set of KEYs $\bar{k}=\{17,23,29,41,47,31,37,19\}$ are collected and associated with the set of OPB $\bar{b}=\{3,3,3,12,12,8,8,8\}$. We can rewrite these two arrays in a simplified form $\{\{k_1,b_1\},\{k_2,b_2\},...,\{k_n,b_n\}\}$ respectively. Using the *Chinese Remainder Theorem (CRT)* algorithm, the two arrays $\bar{k}$ and $\bar{p}$ are represented as a scalar *MCID(s,d)*, it is possible to restore each next-hop information p_i by using corresponding k_i at each intermediate switch (details of these operations are described in [5]). That's to say

$$MCID(s,d)=\mathrm{CRT}\left(\{k_1,b_1\},\{k_2,b_2\},...,\{k_n,b_n\}\right). \qquad (4)$$

In mathematics, the conditions are satisfied if and only if $MCID(s,d)\equiv b_1(\mathrm{mod}\ k_1)\equiv b_1(\mathrm{mod}\ k_2)\equiv ...\equiv b_n(\mathrm{mod}\ k_n)$ for real-valued field. Finally, we obtain the MCID(*s,d*)= 398,117,880,130 through a concrete instance by calculating eq.(6).

Let's take a look in detail how a COXMP packet flows through an intermediate switch. Each incoming multicast packet is in-need duplicated and forwarded to next-hop neighbor switches indicated by the remainder at each intermediate switch *i*, which is constructed by using a simple modulo operation: Let MCID*(s,d)* be the dividend and node-specific key k_i be the divisor. Through a long integer divider, the desired p_i for each intermediate switch *i* is obtained by the remainder p_i. The forwarding logic is given by

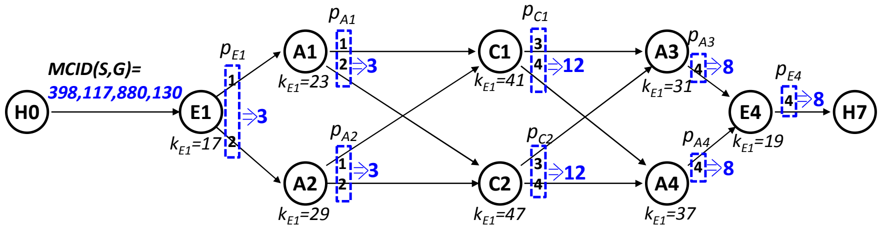

Fig. 1. An example for COXMP forwarding along the reference network.

$$b_i = MCID(s,d) \bmod k_i \tag{5}$$

Now the intermediate switches obtain complete information of all possible paths (interfaces). The packet are thus forwarded to any p_k in which $\forall b_i \wedge 2^k = 1$, where $k = 0,1 \ldots \varepsilon(i)-1$ and $\varepsilon(i)$ denotes the number of ports of switch *i*. There are lots of strategies to determine an exact next-hop interface from an OPB p_i, such as 1) by random; 2) by round-robin; and 3) by network states (e.g., interface speed and output buffer occupancy thresholds etc). The possibilities for manipulating the select strategies do not end there. In this paper, we compared COXMP with round-robin (COXMP-RR) and with congestion avoidance (COXMP-CA) against schedulers commonly used in practice and research, including the LIPSIN and source-route. In COXMP-RR strategy, a COXMP enabled switch detects congestion by monitoring the occupancy of its input or output buffers. When an output buffer's occupancy exceeds a certain threshold (also called instantaneous interface congestion), the switch marks the interface as a congestion interface and preferential forwarding to non-congestion interfaces, thus can maximize the network capacity.

Hence, it can be observed that by having MCID(S,G) and the array k, each element of the array p can be restored in an ECMP(s,d). That is to say, the COXcast packets with different MCID(s,d) and containing different multipath flows' self-routing information traverse a specific intermediate switch, so different OPBs will be obtained from extracting different MCID values with the same key. Therefore one unique key with a different MCID(s,d) is sufficient to distinguish one OPB from another without conflict, and theoretically there are limitless multicast channels. In addition, the order of pair $\{k_i, p_i\}$ is irrelevant.

Illustrated by the example of Fig. 1, source *s* sends the multipath packet flow with COX header to switch *E1*. *E1* receives the packet, obtains the OPB=3 by performing the 398,117,880,130 (mod 17) operation, and forwards the packet either to switch *A1* via port 1, or to switch *A2* via port 2. When switch *E1* selects *A1* as its next hop towards final destination. Switch *A1* receives the packet, obtains the OPB=3 by performing the 398,117,880,130 (mod 23) operation, and forwards the packet to possible next hops *C1* or *C2* via port 1 or port 2 respectively. And so forth, this packet finally arrived at the desired receiver *d*. At each intermediate switch, a result is calculated using a simple modulo operation on the MCID with its own node-specific keys, which result will be used to designate the correct OPBs in each intermediate switch. Since the proposed scheme does not require table lookup and header modification for incoming packets, it is important that processing in the intermediate switches should be stateless, fast and highly scalable in the number of multicast receivers. Thus we can expect that our scheme provides better performance than most of existing schemes.

III. Simulation and Results

We used the Fat-tree DCN topology to simulate the behavior. Using ns-2 [7][8], we implemented a traffic management module and stateless packet-based forwarding which computes forwarding port based on involved links; the selected tree is always defined by the equal cost multi-paths between the publisher and each of the subscribers. According to our previous work [5], we encode all links of the tree into a tag field. Thus, we set the size of tag to 3 bytes in *FT(2,4,16)* and 43 bytes in *FT(8,16,96)* which is compatible with both IPv4 and IPv6 (Note that the overhead of COXMP and COXcast is not equivalent since the number of involved routers differs). The representation of the tag in packet headers is source specific and the tags are very likely different for reaching the same subscriber sets from different sources.

The most important modification of the base forwarding method is that the node needs to be able to determine on which forwarding port it should send among matching candidates. The port selection strategies include Round Robin (COXMP-RR) and our proposed congestion avoidance (COXMP-CA). The mechanism of COXMP-RR is similar to ECMP without causing any additional operation delay. To observe the metrics of congestion-aware routing, we have implemented packet-level multi-path routing in ns-2. Moreover, COXMP-CA takes the status of queue size into consideration and forward packets to the link which has the shortest queue occupancy. However, it will cause slightly extra delay while looking for the optimal outgoing link.

We examine the throughput by allocating the data flow from one node to other host. As before, in these simulations, ECMP is allowed to split traffic over 2^3 equal-cost paths under FT(2,4,16) and 5^3 equal-cost paths under FT(8,16,96). Basically, all of the routes in ECMP can be derived from COXMP or LIPSIN tags. Nevertheless, with Bloom filters, matching may result in some false positives, thus the packet may be forwarded along a link that was not added to the zFilter and it will cause extra traffic which may result in the effective throughput low. In contrast, COXMP can achieve 100% accuracy rate when derived from tag. Besides, the tag size of COXMP is smaller than LIPSIN in tolerable false positive rate which is under 10^{-2}. Hence, we expect that our scheme to provide better performance than most of existing schemes.

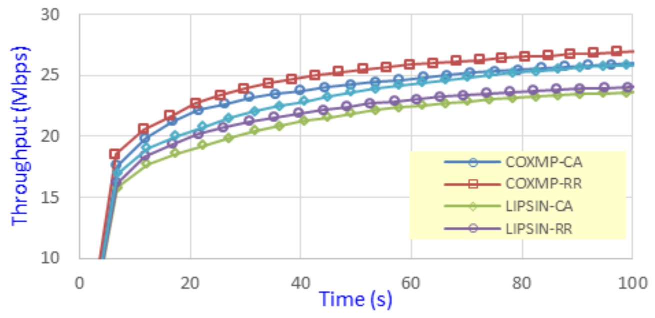

Fig. 2. Throughput of *FT(2,4,16)*.

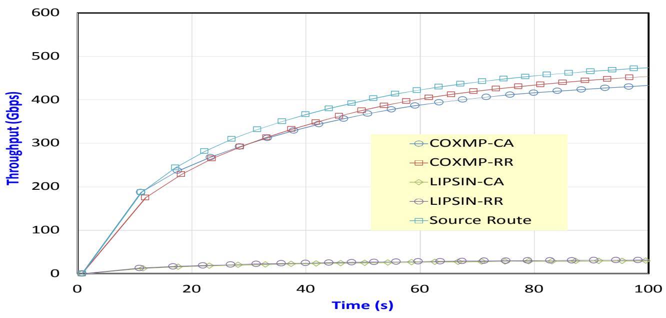

Fig. 3. Throughput of *FT(8,16,96)*.

Note that the throughput in our comparative analysis is effective throughput, i.e., we summarize the payload of each packet.

Fig. 2 and 3 give the network throughput comparison of different routing schemes under various traffic flows. First, we observe that the system throughput of COXMP is higher than LIPSIN. This is because the overhead of COXMP in *FT(2,4,16)* and *FT(8,16,96)* are 57 and 97 bytes individually and the overhead of source route is 82 bytes. According to the FPR described in [6], we found that the {m, n, k} which satisfies our FPR requirement are {1064,112,7} in *FT(2,4,16)* and {6647,700,7} in FT(8,16,96). The effective throughput is 27.93Gbps in COXMP-RR, 27.05Gbps in source route and 25.06Gbps in LIPSIN-RR under *FT(2,4,16)*. The COXMP-based multipath routing has 3.2% improvement in comparison with source route in small scale. The effective throughput is 492.28Gbps in COXMP-RR, 510.97Gbps in source route and 33.76Gbps in LIPSIN under *FT(8,16,96)*. The COXMP-based multipath routing has performance degradation in comparison with source route in large scale case. Obviously, the influence of overhead in COXMP and LIPSIN is tremendous especially in large scale condition since the length of tag involves the number of routers in COXMP and links in LIPSIN. This is also the reason why COXMP always outperforms LIPSIN as the FPR is less than 0.01.

Fig. 4 depicts the UDP throughput comparison under different background traffic intensity in *FT(2,4,16)*. It is not surprising that COXMP still performs the best. This is because, with light overhead, COXMP could get more improvement because the packet size is small. Taking the audio data compression G.729 as an example, the size of payload is 10 bytes in each packet. That is, in comparison with payload, the proportion of overhead is relatively high (e.g. the packet size of COXMP in G.729 is 67 bytes which includes 14 bytes for Ethernet header, 20 bytes for IP header, 8 bytes for UDP header, 12 bytes for RTP header, 3 bytes for tag of COXMP and 10 bytes for payload). With the forwarding policy, congestion avoidance (CA) module outperforms Round Robin (RR) since CA can dynamically select a single forwarder candidates, thus it performs better load balance than RR. The effective throughput of G.729 in COXMP-CA is 1.18 Gbps under 1,000 background flows. In comparison with COXMP, the effective throughput of source route and LIPSIN are 0.9 Gbps and 0.42 Gbps respectively. COXMP achieve 31.1% improvement than source route in *FT(2,4,16)*. Clearly, the throughput of UDP is much lower than TCP since the overhead ratio in real-time application is critical.

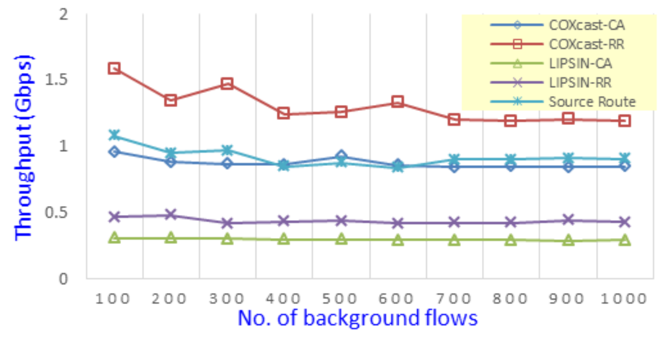

Fig. 4. Throughput of UDP flows with background traffic under *FT(2,4,16)*.

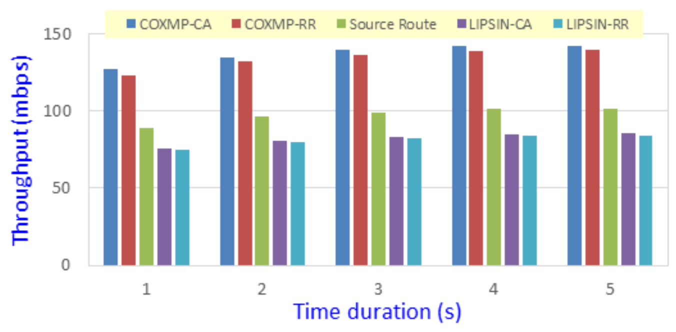

Fig. 5. Background traffic with single link failure.

In a single link failure scenario, we want to observe the impact of single link failure. As shown in Fig. 5, it demonstrates that for any given approach, CA remains the best performance. This is because once the single link failure occurs, the traffic-aware management module would notify the involved publishers and reproduce the new tag by publisher router. The time complexity of COXMP is $O\left(n \times \log(M_{cp})\right)$ where M_{cp} is the consecutive product of involved link IDs and n denotes the number of unicast and multicast flows, and of LIPSIN is $O(k)$ where k denotes the number of hash functions used by LIPSIN. Thus, the influence of single link failure on congestion avoidance is trivial. In contrast, the RR-based scheme would reproduce the new tag until the next joining/leaving query arrives; source route needs to regenerate the routing paths and the time consumption depends on corresponding routing algorithms. Here we use the Link-State routing algorithm and the time complexity of mentioned schemes are small so that it would not cause bottleneck while producing new routes.

IV. Conclusions

The forwarding scheme is generally regarded as a critical part in the datacenter network in which processor, memory, and I/O devices are dynamically shared. Network performance and reliability are key design goals, but they are tempered by cost and serviceability constraints. On the other hand, a good user experience relies on predictable performance, with the datacenter network of predictable latency and bandwidth characteristics across varying traffic patterns. The traditional single-path routing has exposed bottlenecks in the DCN that require better multipath routing algorithms and forwarding schemes for efficient data delivery. Consequently, many researches have sought to improve the overall performance of multipath forwarding for the datacenter networks. In this paper, we introduce a novel source and stateless multipath forwarding scheme in datacenter networks for reducing the overhead by compressing the source multipath routing information into incredibly small size in a way that increases the performance and scalability of the proposed scheme. Compared to BF-based multipath forwarding schemes, COXMP reduces the processing cost, protocol overhead and delivery latency, while simplifying the deployment and management of a DCN with large number of alterative routing paths, especially when applied to large-scale DCNs.

REFERENCES

[1] L. Kleinrock. Queueing Theory. Wiley, New York, 1975

[2] V. Arya, T. Turletti, and S. Kalyanaraman, "Encodings of multicast trees," Lecture Notes in Computer Science, vol. 3462, pp. 992–1004, May 2005.

[3] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast Security: A Taxonomy and Some Efficient Constructions," Proc. IEEE INFOCOM, pp. 708-716, 1999.

[4] A. Perrig, D. Song, and D. Tygar, "ELK, a New Protocol for Efficient Large-Group Key Distribution," Proc. IEEE Symp. Security and Privacy, pp. 247-262, 2001.

[5] W.-K. Jia, "A Scalable Multicast Source Routing Architecture for Data Center Networks," IEEE Journal on Selected Areas in Communications, vol.32, no.1, pp.116,123, January 2014

[6] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: line speed publish/subscribe inter-networking." SIGCOMM Comput. Commun. Rev. 39, 4, August 2009

[7] S. McCanne and S. Floyd. ns Network Simulator. http://www.isi.edu/nsnam/ns/.

[8] T. Issariyakul and E. Hossain, "Introduction to network simulator ns2" Springer, Nov. 2008.

[9] "Software-Defined Networking: The New Norm for Networks". White paper. Open Networking Foundation on April 2012

[10] A. Greenberg, J. Hamilton, N. Jain et al., "VL2: A scalable and fl exible data center network," in Proc. ACMSIGCOMM,Aug. 2009, pp. 51–62.

[11] C. Guo, G. Lu, D. Li et al., "BCube: A high performance, server-centric network architecture for modular data centers," in Proc. ACM SIGCOMM, Aug. 2009, pp. 63–74.

[12] A. Greenberg, J. Hamilton, D. Maltz et al., "The cost of a cloud: Research problems in data center networks," in Proc. SIGCOMM Comput. Commun. Rev. (CCR), 2009, vol. 39, no. 1, pp. 68–73.

[13] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in Proc. ACM SIGCOMM, Aug. 2008, pp. 63–74.

[14] D. Li, C. Guo, H. Wu et al., "Scalable and cost-effective interconnection of data center servers using dual server ports," IEEE/ACM Trans. Netw., vol. 19, no. 1, pp. 102–114, Feb. 2011.

[15] S. Kandula, S. Sengupta, A. Greenberg et al., "The nature of datacenter traffic: Measurements & analysis," in Proc. ACM SIGCOMM Internet Meas. Conf. (IMC'09), Nov. 2009, pp. 202–208.

[16] T. Benson, A. Anand, A. Akella et al., "Understanding data center traffic characteristics," in Proc. Workshop Res. Enterprise Netw. (WREN'09), 2009, pp. 65–72.

[17] S. Floyd, V. Jacobson, S. McCanne et al., "Reliable multicast Framework for light-weight sessions and application level framing," in Proc. ACM SIGCOMM, Oct. 1995, pp. 342–356.

[18] S. Paul, K. Sabnani, J. Lin et al., "Reliable multicast transport protocol (RMTP)," IEEE J. Sel. Areas Commun., vol. 15, no. 3, pp. 1414–1424, Apr. 1997

[19] J. Griffioen and M. Sudan, "A reliable dissemination protocol for interactive collaborative applications," in Proc. ACM Multimedia, Nov. 1995, pp. 333–344.

[20] H. Holbrook, S. Singhal, and D. Cheriton, "Log-based receiverreliable multicast for distributed interactive simulation," in Proc. ACM SIGCOMM, Oct. 1995, pp. 328–341.

[21] T. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees (CBT): An Architecture for Scalable Inter - Domain Multicast Routing, " in Proceedings of ACM SIGCOMM'93, pp. 85 - 95, 1993.

[22] D. Waitzman, C. Partridge, S. E. Deering, "Distance Vector Multicast Routing Protocol", Internet Request for Comment 1075, November 1988.

[23] A. S. Thyagarajan, and S. E. Deering, "Hierarchical Distancevector Multicast Routing for the MBone," in ACM SIGCOMM Computer Communication Review, pp.60 - 66, 1995.

[24] A. Detti, N. Blefari Melazzi, S. Salsano, and M. Pomposini, "CONET: a content centric inter-networking architecture," in Proc. of the ACM SIGCOMM ICN Workshop, 2011, pp. 50–55.

[25] T. Speakman, J. Crowcroft, J. Gemmell et al., "PGM reliable transport protocol specification," RFC3208, Dec. 2001.

[26] L. Lehman, S. Garland, and D. Tennenhouse, "Active reliable multicast," in Proc. IEEEConf. Comput.Commun. (INFOCOM'98),Mar. 1998.

[27] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in Proc. of the ACM CoNEXT, 2009, pp. 1–12.

[28] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," Computer Communications, vol. 36, no. 7, pp. 779 – 791, 2013.

[29] D. Perino and M. Varvello, "A reality check for content centric networking," in Proc. of the ACM SIGCOMM ICN Workshop, 2011, pp. 44–49.

[30] H. Yuan, T. Song, and P. Crowley, "Scalable NDN forwarding: Concepts, issues and principles," in Proc. of the IEEE ICCCN, 2012, pp. 1–9.

[31] A.T. Sherman and D.A. McGrew, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees," IEEE Trans. Software Eng., vol. 29, no. 5, pp. 444-458, May 2003.

[32] D.M. Wallner, E.J. Harder, and R.C. Agee, "Key Management for Multicast: Issues and Architectures," Internet Draft, Internet Eng. Task Force, 1998.

[33] G. Caronni, K. Waldvogel, D. Sun, and B. Plattner, "Efficient Security for Large and Dynamic Multicast Groups," Proc. IEEE Seventh Int'l Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 376-383, 1998.