A Scalable Multicast Source Routing Architecture for Data Center Networks

Wen-Kang Jia, *Member, IEEE*

***Abstract*—This paper introduces a new scalable, efficient and stateless source routing scheme to ensure that unified unicast and multicast packets can be delivered in the Data Center Networks (DCNs). This scheme, called Code-Oriented eXplicit multicast (COXcast), is based on unicast and eXplicit multicast (Xcast). It constructs the unicast path and multicast tree by encoding the corresponding output port bitmap of each intermediate node. Using a common identifier and a node-specific key, so packets can be self-routed to multiple receivers without requiring header modification. In addition, intermediate switches/routers on the path/tree can be stateless. Compared to traditional source-based unicasting and explicit-based multicasting schemes, COXcast has lowered the processing cost, protocol overhead and delivery latency, while simplifying the deployment and management of a large number of medium-scale multicast groups, especially when applied to large-scale DCNs.**

***Index Terms*—Code-Oriented eXplicit multicast (COXcast), eXplicit multicast (Xcast), source routing, Chinese Remainder Theorem (CRT), Data Center Networks (DCNs).**

I. Introduction

THE DEMANDS for multicast services for network applications characterized by one-to-many or many-to-many dissemination have been growing rapidly. These services include multiparty audio call, video conferencing, whiteboard, push to talk, live streaming services, collaborative applications, e-learning, on-line gaming, distributed interaction, software upgrading, distributed database replication, etc [1]. Multicasting can benefit group applications by reducing network traffic, saving communication costs, improving application throughput, as well as by supporting efficient group collaboration in cloud computing applications. These improvements are most significant for cloud *Data Center Networks (DCNs)*.

Traditional multicast [2]–[5] schemes are based on managing one-to-many networks resources efficiently. While traditional multicast schemes are scalable for very large multicast groups, they have scalability problems for large distinct multicast groups. Since some multicast services are required only for the close collaboration of small teams, eXplicit multicast (Xcast) [6] was proposed as a new multicast category with complementary scaling properties. It achieves this by encoding the list of destinations explicitly in the packet header, instead of using a multicast group address. Thus, based on existing unicast routing information, the intermediate router can be stateless for forwarding, instead of relying on multicast routing information. Xcast can efficiently support a very large number of distinct small multicast groups and thus can play an important role in making multicast applications applicable for small groups. However, Xcast brings up a new scalability issue: due to the oversized multicast packet header, both the group size and payload size in Xcast are restricted. Both categories of multicasting, therefore, are unscalable on either number of groups or number of receivers, and solution to this scalability problem of multicasts has been proved to be NP-complete [7].

To solve the major problems of datacenter multicasting, especially those of efficiency and scalability, we propose a new approach based on the Xcast scheme, called *Code-Oriented eXplicit multicast (COXcast)*. As the name implies, COXcast is similar to Xcast, in that all the destinations information in the packet header are tagged. Through appropriate algorithm, the on-tree forwarding information is encoded into an identifier in the multicast packet header, which is resolved into an output port bitmap by a node-specific key at each traversed router. Thus proposed scheme is similar to a packet staffed with a 'cox', so that it can be self-routed toward all the receivers. Our contribution is an older algorithm for efficient solving the deployment issues of the datacenter multicast services.

The rest of this paper is organized as follows: we review the related work and discuss related issues in Section II. In Section III, we describe the proposed scheme in detail. In Section IV, we evaluate the performance of proposed scheme, compare its strengths and limitations to previous works, and present simulation results. Finally, we conclude the paper in Section V.

II. Datacenter Multicasting and its Issues

Multicast is originally a key component for effective scaling within and between DCNs' overlay applications. For instance, the majority of the cooperative applications are discovering and signaling each other, so there may be better ways to support them then using layer-2 multicast. However, the multicasting of data is occurring in large volumes, so there is a need for good layer-3 multicast support.

Small group applications [8], [9] such as multiparty conferencing is currently highly desired DCN applications. Multicasting is expected to help conserve bandwidth in the DCNs and reduce the load on server farms. The communication model for small-group applications differs from conventional unicast and traditional multicasts. For example, in an *N*-participant conferencing, each participant sends its statement or media stream to the others using multicast directly. Here that each participant acts as a source that has *N-1* receivers, and states of maximum *N* multicast channels are created in each on-tree router. If there are *M* conferences taking place simultaneously,

Manuscript received January 15, 2013; revised July 1, 2013.

W.-K. Jia is with the Department of Computer Science, National Chiao Tung University, Taiwan (e-mail: wkchia@cs.nctu.edu.tw).

Digital Object Identifier 10.1109/JSAC.2014.140111.

0733-8716/14/$31.00 © 2014 IEEE

$M \times N$ multicast groups are there, such application thereby may have a big scalability problem for the number of groups because it uses traditional multicast protocols. On the other hand, for highly available datacenter storage systems, every disk volume is typically replicated on a multi-way mirror system either locally or remotely, this N-way data replication increases the redundant traffic load on the network links by N times. In order to reduce the latency associated with the maintenance of data replica consistency, multicast is a natural solution to this systems [10]. Consider that a practical replication level is less than or equal to 8 ways, thus Xcast is also the best choose for such applications. Since small group applications are usually limited in the demand of number of participants. Compare to traditional multicast, the Xcast is inherently more suitable solution for provisioning few-to-few services to such applications with massive small-groups [11].

As mentioned before, the consequences of such increases demand for application characteristics has changed how applications are currently hosted in datacenters. To respond to one of the most important challenges for the DCNs, it's important to understand how multicast properly supports these applications in the continually evolving DCNs. In order to overcome the scalability problems in terms of multicast group sizes, Xcast has experienced various enhancements such as Xcast+ [12], GXcast [13], SGM [9], MSC [8], REUNITE [14], and HBH [15]. The majority of these propositions tried to reduce the overhead by using hybrid stateful and stateless approach, but none of them have really solved the scalability problems yet. Hence, we find the technical trends of modern DCN design have brought up new challenges for multicasting:

First, in order to fairly and effectively use this bandwidth requires ensuring traffic flows achieve the maximum throughput, the topologies of DCNs usually lead to high link density and redundancy. For example, recently DCN topologies such as Fat-Tree [16], BCube [17], Portland [18] and VL2 [19] typically provide many equal-cost paths between arbitrary given pair of hosts with multipath routing. In this scenario, multicast trees formed by traditional independent receiver-driven multicast routing can result in severe link waste compared to efficient ones. In other words, enabling multipath support in DCNs is necessary for multicast traffics. Previous investigation shows that source routing can achieve this goal by establishing multiple path pairs through source control [16], [17], but it is not easy to represent a multicast tree by a cost-efficient encoding method [20], which poses a challenge.

Second, entry-level commercial switches are widely used in most DCN designs for economic and scalability considerations, through the memory space of their multicast forwarding/routing table are relatively narrow, less than 1,500 entries [21]. Thus performance of the multicast forwarding states lookup operation could be the bottleneck. Furthermore, it is difficult to aggregate in-switch multicast routing entries since the multicast address embeds no topological information. Hence, it is quite challenging to support the many thousands of multicast groups in DCNs.

Third, the fundamental feature of stateful multicast is that the routers are responsible for every group's initial establishment by contacting involved routers. The convergence time and signaling overhead of stateful multicast routing protocols are inefficient for multicast tree construction. The problem becomes much more severe for multicast groups with many members and high member join-leave frequency. In order to support the emerging and demanding requirements of cloud applications, datacenter multicasting needs to perform beyond the current requirements for QoS, reliability, availability, resilience, load-balance, etc. At the same time, facing the limit of the commercial deployment for Internet-scale multicasting, the closed and well-managed DCNs can also provide a challenging opportunity for pioneered multicast deployment.

Fourth, the stateless-approach multicast protocols such as Xcast also have inscalability and inefficiency problems as in non-DCNs. The effective payload of Xcast packet will be reduced if the number of receivers grows, and this problem cannot be solved by packet fragmentation [1]. Since packets are bounded by MTU in size, Xcast faces a tradeoff between group size and payload size. The another drawback of Xcast is the forwarding cost: an Xcast router must send multiple copies of Xcast packet with different header content; hence routing lookup and header recombination processes may seriously reduce the performance of on-tree routers, not only processing time but also memory space are occupied, especially in the branching routers. The problem becomes much severe for the high link density of DCNs' topologies. In addition, the switch/router will spends too much time on the unicast table lookup, separating and combining the list of destination addresses, which will cause massive and uncertain delivery latency and hurts some real-time applications.

Deploying successful multicast services relies on the effective underlying multicast enabled networks. In spite of the benefits from multicast technologies and continuous improvement of multicast routing protocols, numerous restrictions still exist, especially in scalability and flexibility. Through proposed scheme, these restrictions of multicast should be lifted within the next several years.

III. Code-Oriented Explicit Multicast

We have propose a novel multicast routing architecture to deal with various existing issues regarding in today's DCNs. In proposed scheme, the considered DCNs are characterized entirely by COXcast-enabled switches, and the scheme can be adopted to *Software Defined Networking (SDN)* framework [22] in the future. Our goals are to improve the scalability ans efficiency performance of the current multicast routing protocols, and provide higher flexibility in the deployment of new multicast services in a great quantity of medium-scale groups with few thousands of participants.

A. Basic Concepts

There are three key elements for the COXcast: 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. Based on the properties of the CRT, a MCID divided by different KEYs will remain as different OPBs as a remainder, that's to say $\text{MCID} \equiv \text{OPB}_1 (\text{mod } \text{KEY}_1) \equiv \text{OPB}_2 (\text{ mod } \text{KEY}_2) \equiv$

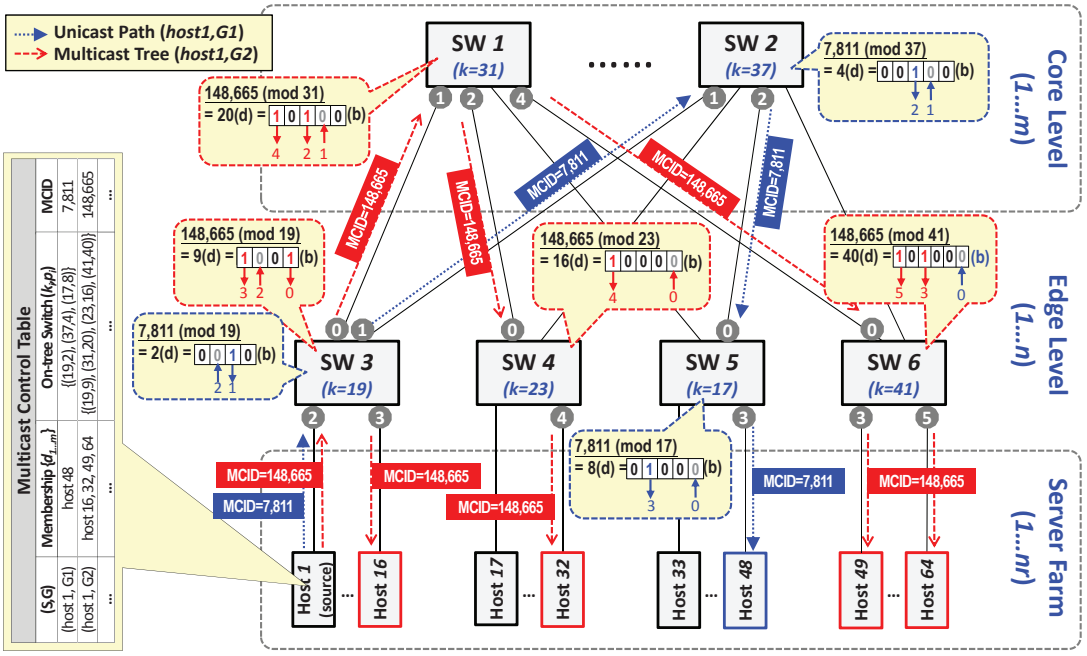


Fig. 1. COXcast operations in the *FT*(m=2,n=4,r=16) DCN.

$\ldots \equiv \text{OPB}_n$ (mod KEY_n). We use this property to represent the entire multicast delivery tree in a DCN.

At the source (sender), the KEYs and OPBs through the optimal (shortest) path and further multicast tree in the network are collected by inter-operating with each on-tree intermediate switch, using appropriate protocols. In order to support the COXcast, each source should maintain a data structures called *Multicast Control Table (MCT)*. A multicast group (channel) in COXcast is identified by *(S,G)* according to [23], each active multicast group *(S,G)* it contains a list of participating receivers and their current states. The table also contains KEYs and the associated OPBs of all on-tree switches (*Forwarding Set(FS)*) for each *(S,G)*. Using MCT, the source can construct the MCID for each unicast path as well as multicast tree. Figures 1 illustrates the MCT. Once the group membership changes, the corresponding entry should be updated. Due to lack of topology knowledge at the source in case of SDN-based DCNs, the SDN controller could offer a query interface to provide the information toward the source. Furthermore, an arithmetic of MCID generation is performed based on the CRT) routing [24]. The derived MCID is an in-packet message placed in a special header extension (COXcast header) in either IPv4 or IPv6 multicast packets, and that MCID must be processed by every switch along the unicast path or multicast delivery tree hop by hop.

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. Details of these operations are described in the following subsections. 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 to provide better performance than mostly existing schemes.

The basic operation of the COXcast is illustrated in Figure 1, in which a multicast packet traverses through a reference DCN. The DCN is a fat-tree which has been widely adopted as the topology for DCNs, and it can be modeled as a notation *FT*(m,n,r) to denote such a fat-tree: the topology of a two-level fat-tree, where each edge in the fat-tree network consists of bidirectional links between two neighboring nodes, given the number of edge switches (lower-level) n and the number of nodes associated with each edge switch *r*, which consists of an *r*-port downlink and *m*-port uplink configuration. The connection capacity of a fat-tree network *FT*(m,n,r) depends on the number of core switches (upper-level) *m*, which consist of *n*-port downlink configurations. Thus we have $n \times r$ hosts and $n \times (2m+r)$ switch ports in the network.

There are four major steps for COXcast operation: 1) topologic discovery at source and key assignment at intermediate switches; 2) membership management such as joining and leaving between receiver(s) and source; 3) MCID calculation at source, and 4) packet delivery from source to receiver(s) through intermediate switches. The COXcast supports two types of packets: control and data packets, the latter may contain a MCID field, and the former contains several type of messages such as JOIN and LEAVE.

On the other hand, to ensure that each switch obtains a unique key, and that all the keys in the network are pairwise relative primes, and each of the keys must be larger than the port bitmap size ($2^{\text{num. of ports}}$) of the switch. Either static or dynamic manners should be deployed, so when the DCN is configured, each DCN switch can be pre-assigned a unique key value that satisfies the requirements previously described. However, this is costly and error prone, so using dynamic assignment may be a better choice. In this approach, we can

deploy a centralized key management infrastructure similar to DHCP. Here, a COXcast key management server is adopted in the system design, collects key request, generates keys, and assigns keys automatically to the requesting switches.

B. MCID Generation Arithmetic at Source

In order to construct a MCID of *(S,G)*, the information of multiple unicast paths that needs to be collected by source first, the underlying topology discovery protocol performs the path information gathering procedure that involves all forwarded nodes along the end-to-end path

$$\begin{aligned}\mathbf{P}\left(S \to d_s\right) &= \left\{ \left(k_{x,1}, \varepsilon\left(k_{x,1}\right)\right), \left(k_{x,2}, \varepsilon\left(k_{x,2}\right)\right), \ldots \right. \\ &\qquad \left. , \left(k_{x,\ell(x)}, \varepsilon\left(k_{x,\ell(x)}\right)\right) \right\} \\ \Rightarrow \quad \mathbf{P}(S \to d_s) &= \left\{ \left(k_{x,i}, \varepsilon\left(k_{x,i}\right)\right) \middle| \begin{array}{c} k_{x,i} \in \overline{k_x} \subset \mathbb{P}, \\ \varepsilon\left(k_{x,i}\right) < k_{x,i} \text{ and } \varepsilon\left(k_{x,i}\right) \in \overline{\varepsilon_x} \\ \text{, where } d_x \in G, x = 1, 2, \ldots, m \\ \text{, and } i = 1, 2, \ldots, \ell\left(x\right) \end{array} \right\} \end{aligned} \tag{1}$$

of receiver d_x with path length $\ell(x)$, where k_x denotes the node specific key of switch x, which is positive integer and pair-wise co-prime with each other $k_{i \neq x}$, and $\varepsilon(k_x)$ denotes its corresponding *Output Port Index (OPI)* and maximum $\varepsilon(k_x)$=$2^{\upsilon_x} < k_x$,where υ_x denotes number of ports of switch x. With multiple unicast paths, a multicast tree consists of all the forwarded nodes can be represented by recursively merging these two arrays $\overline{k_x}$ and $\overline{p_x}$:

$$\mathbf{T}(S \to G) = \left\{ (k, p(k)) \middle| \begin{array}{c} k \in \bigcup_{x=1}^{m} \overline{k_x}, \\ \begin{cases} p(k) = \varepsilon(k_{x,i}) & , \exists k_{x,i} \notin \bigcap_{x=1}^{m} k_{x,i} \\ p(k) = \bigwedge_{x=1}^{m} \varepsilon(k_{x,i}) & , \text{otherwise} \end{cases} \end{array} \right\}, \tag{2}$$

where $G = \{d_1, d_2, \ldots, d_m\}$ denotes the destination set, $x = 1, 2, \ldots, m$ and $i = 1, 2, \ldots, \left|\bigcup_{i=1}^{m} \overline{k_i}\right|$.

We can rewrite the Eq.(2) in simplify form $\overline{k}$=$\{k_1, k_2, \ldots, k_n\}$ and $\overline{p}$=$\{p_1, p_2, \ldots, p_n\}$ respectively, where n=$\left|\bigcup_{i=1}^{m} \overline{k_i}\right|$ denotes the tree size. These two arrays are denoted by the keys and the desired OPBs (merged from multiple OPIs) also known as FS of (S, G)=$\{\overline{k}, \overline{p}\}$ of the designated unicast path and multicast tree.

The MCID can be implemented straightforwardly using the CRT with $\overline{k}$ and $\overline{p}$. First, a new scalar $\mathbf{K}$ is defined as a continued product of all n elements in the array $\overline{k}$:

$$\mathbf{K} = \prod_{i=1}^{n} k_i. \tag{3}$$

Using $\mathbf{K}$ and $\overline{k}$, we create a new array

$$\overline{m} = \{m_1, m_2, \ldots, m_n\} = \{\frac{\mathbf{K}}{k_1}, \frac{\mathbf{K}}{k_2}, \ldots, \frac{\mathbf{K}}{k_n}\}. \tag{4}$$

Another array $\overline{c} = \{c_1, c_2, \ldots, c_n\}$ is created based on the arrays $\overline{m}$ and $\overline{k}$

$$c_i = m_i \times \left(m_i^{-1} (\mathrm{mod}\, k_i)\right), \forall i \leq n. \tag{5}$$

The term m_i^{-1} denotes the multiplicative inverse of $m_i \bmod k_i$ defined by $m_i^{-1} m_i (\mathrm{mod}\, k_i) = 1$. This requires that m_i and n_i should have no common divisors larger than 1, i.e. $\gcd(m_i, k_i) = 1$. Because m_i is the product of all n-elements minus k, all the n-elements should be relative primes to satisfy the requirement. The scalar MCID*(S,G)* can be easily calculated using array $\overline{p}$ and $\overline{c}$

$$\mathrm{MCID}(S, G) = \left(\sum_{i=1}^{n} (p_i \times c_i)\right) (\mathrm{mod}\, \mathbf{K}). \tag{6}$$

For a MCID*(S,G)*, let n be among of FS in a multicast delivery tree or unicast path. The time complexity to construct a MCID is written as $\mathcal{O}\left(n \times \log \mathbf{K}\right)$. The average bit length of MCID*(S,G)* (say space complexity) can be derived as

$$|\mathrm{MCID}(S, G)| = \frac{\left((\mathbf{K} - 2^{\gamma}) + \sum_{i=0}^{\gamma-1} \left(2^i\right)\right)}{\mathbf{K}}, \tag{7}$$

where $\gamma = \lfloor \log_2 \mathbf{K} \rfloor$, which known as the asymptotic law of distribution of prime numbers. Thus the average bit length of $\mathbf{K}$ can be expressed as

$$|\mathrm{MCID}(S, G)| \leq |\mathbf{K}| \approx \sum_{i=\rho+1}^{\varphi+1} \left[\left(\pi(2^i) - \pi(2^{i-1})\right) \times i\right], \tag{8}$$

where $\pi(2^{\varphi}) \leq (n + \pi(2^{\rho}))$, and $\pi(x) = \frac{x}{\ln x}$ denotes the prime-counting function that gives the number of primes less than or equal to x, for any real number x.

C. Packet Delivery at Intermediate Switches

Let's take a look in detail how the COXcast 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,G)* be the dividend and node-specific key k_i be the divisor. Through a long integer divider, the desired $\widehat{p_i}$ for each intermediate switch i is obtained by the remainder p_i. The forwarding logic is given by

$$p_i = \mathrm{MCID}(S, G) (\mathrm{mod}\, k_i). \tag{9}$$

Hence, it can be observed that by having MCID*(S,G)* and the array $\overline{k}$, each element of the array $\overline{p}$ can be restored in a unicast path $\mathbf{P}(S \to d)$ or multicast tree $\mathbf{T}(S \to G)$. That is to say, the COXcast packets with different MCID*(S,G)* contain different channels' self-routing information traverse a specific intermediate switch, and so different OPBs will be obtained from extracting different MCID values with the same key. Therefore one unique key with a different MCID is sufficient to distinguish one OPB from another with no conflict, and theoretically there can be limitless multicast channels. In addition, the order of pair $\{k_i, p_i\}$is irrelevant.

D. Example of COXcast operation

In the following we provide two examples of unicast and multicast operations using COXcast. In the case of unicast forwarding in Figure 1, when a source node $host1$ wants to establish a connection toward a destination node $host48$, we have the unicast path sequence $\{SW3, SW2, SW5\}$. Based

on the proposed arithmetic, three KEYs $\{19, 37, 17\}$ are created and associated with OPIs $\{1, 2, 3\}$, which is transferred to OPBs $\{2(00010b), 4(00100b), 8(01000b)\}$. Finally, we obtain the MCID*(host1,G1)*=7,811 by calculating CRT arithmetic. Note that the order of key and OPB entries is irrelevant. Now $host1$ sends the unicast packet flow (The blue dotted line) with COXcast header to $SW3$. $SW3$ receives the packet, obtains the OPB=2 by performing the $7,811(mod\,19)$ operation (detailed as in Eq.9), and forwards the packet to $SW2$ via port 1. $SW2$ receives the packet, obtains the OPB=4 by performing the $7,811(mod\,37)$ operation, and forwards the packet to $SW5$ via port 2. $SW5$ receives the packet, obtains the OPB=8 by performing the $7,811(mod\,17)$ operation, and forwards the packet to final destination $host48$ via port 3.

In the case of multicast forwarding, when source $host1$ has received the JOIN requests from $host16$, $host32$, $host49$, and $host64$, a source-route multicast tree is established by its associated source and group ID $(host1, G2)$, and a multicast forwarding set $\{SW1, SW3, SW4, SW6\}$ can be found. Then several KEYs $\{31, 19, 23, 41\}$ are created and associated with OPIs set $\{\{2, 4\}, \{0, 3\}, \{4\}, \{3, 5\}\}$, which are transferred to OPBs 20, 9, 16, 40. Finally we obtain a MCID*(host1,G2)*=148,665 by calculating CRT arithmetic. Then $host1$ sends the multicast packet(s) (The red dotted line) with COXcast header to $SW3$. $SW3$ obtains the OPB=9 by $148,665(mod\,19)$, and forwards the packet to $SW1$ via port 0 and to $host16$ via port 3 respectively. $SW1$ receives the packet, obtains the OPB=20 by $148,665(mod\,31)$, and forwards the packet to $SW4$ and $SW6$ via ports 2 and 4 respectively. $SW4$ obtains the OPB=16 by $148,665(mod\,23)$, and forwards the packet to destination $host32$ via port 4. $SW6$ obtains the OPB=40 by $148,665(mod\,41)$, and forwards the packet to destination $host49$ and $host64$ via ports 3 and 5 respectively. Thus this packet has arrived at all desired receivers.

IV. Performance Evaluation

In this section, we analyze the characteristics and evaluate the performance of COXcast with existing schemes. Compared with previous multicasting schemes, our scheme is a newer paradigm with distinguished characteristics. Table I summarizes the main differences among them. In order to compare the performance of proposed scheme and the selected competitors, we conduct simulations that validate the scalability and efficiency using C, and the quantitative metrics are used to measure the simulation models as following subsections.

Since the *FT* and *BCube* are the most popular DCN topologies [16], [18], we consider both the reference networks needs of our simulations model: 1) *FT*(m,n,r), and 2) *BCube*(n,k). The investigation results from large backbone networks operated by different DCN service providers indicate that *FT*(8,16,96) are sufficient. Therefore our study focuses on network size of less than 128 switches in the reference network. A *BCube*(n,k) network is constructed from $n^k(k+1)$ n-port switches, n^k of them are edge switches. Thus we have n^{k+1} hosts and $n^{k+1}(k+1)$ switch ports, and each host has $k+1$ *Network Interface Cards (NICs)* in the *BCube* network.

In order to indicate the scalability issues of the proposed scheme, a large number of network sizes have been constructed through simulation, including 23 types of *FT* and 16 types of *BCube* networks. COXcast in *BCube* networks have become more complicated because of the COXcast must be enabled in each host. Before the simulation starts, each router must be randomly assigned a unique key via centralize control in the reference network. The one million prime numbers in 24 bits can represent all keys consumed for all COXcast switches, so no more than 64 bits are needed for future extension. Note that the selection of prime numbers depends on the number of interfaces. When a source and its associated receiver(s) are randomly selected from the reference network, the shortest paths connecting each receiver to its source are found, and then we create a multicast tree by merging these unicast paths. The tree's expected value increases as the network size grows and the number of participating receivers' increases. These network models hence reflect a more realistic situation.

A. Protocol Overhead and Scalability

We have experimented with several scenarios and then compared the results with other stateless multicast protocols, such as Xcast4, Xcast6, Xcast4+ and Xcast6+. For the first scenario, a central key management assigns a minimum key to each involved switch in the unicast path or multicast delivery tree, and this key is incremented progressively in the reference network. Similar to Xcast's header, we expect this longer MCID of COXcast can cause a larger overhead, which will reduce the effective payload. The required in-packet overhead can sometimes reflect the scalability. We have observed the scalability from two perspectives: 1) unicast path lengths, and 2) multicast group sizes.

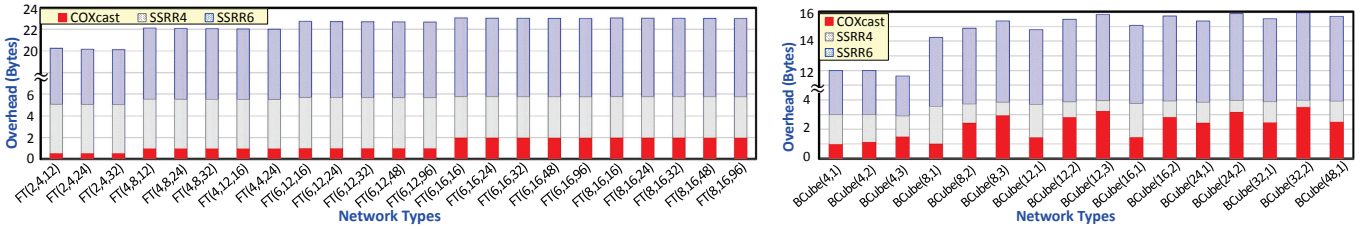
In unicast scenario, the length of a unicast path between two hosts is the hop-count between them when connection is established. We compare COXcast to *Strict Source and Record Route (SSRR)* over IPv4 [25] and IPv6 [26]. We consider that for the various sizes of *BCube* and *FT* networks respectively. We pick the arbitrary 10,000 pair of unicast paths with the minimum hop-counts and then we calculate COXcast header and SSRR header sizes to each of these two paths. COXcast incurred a total overhead of 2 bytes in *FT* networks, and 3 bytes in *BCube* networks. The SSRR4 incurred 4$\sim$6 bytes of overhead and SSRR6 incurred as much as 16$\sim$23 bytes of overhead. COXcast maintained a tight bound on worst case packet overhead while doing on average much better than either SSRR4 or SSRR6 as can be seen in Figure 2(a) and (b). For a 1,500-byte packet the maximum possible COXcast overhead ratio is 0.199%, but in fact in these simulations no packet incurred more than three bytes of overhead. Drastically reducing the protocol overheads of source routing on the transmit path in this way can significantly increase the performance seen by the servers of DCNs. That is because in such network architecture and scalability, the selected keys are usually kept in a low sized range. Besides, a routing path is the minimum hop-count between the two hosts, and the source routing information is compressed into incredibly small size by proposed arithmetic.

The second simulation scenario is focused on the multicast group sizes, and we compare COXcast to Xcast4, Xcast6, Xcast4+ and Xcast6+ for the network sizes of *FT*(2,4,24),

TABLE I
COMPARISON AMONG VARIOUS MULTICAST PROTOCOLS

Protocol	COXcast	Traditional Multicast	Xcast	Xcast+
Routing state in the switch	No	$\mathcal{O}(SG)$	No	No
Control state in the switch	No	$\mathcal{O}(IG)$	No	No (core) / $\mathcal{O}(DSG)$ (edge)
Routing state in the source	No	No	No	No
Control state in the source	$\mathcal{O}(DGR)$	$\mathcal{O}(G)$	$\mathcal{O}(DG)$	$\mathcal{O}(EG)$
Addressing	Unicast+Xcast	Multicast	Unicast+Xcast	Unicast+Xcast+Multicast
Scalability (# of groups)	Huge (∞)	Poor*3 ($\leq 1.5k$)	Huge (∞)	Medium*2 ($\leq 1k$ per edge)
Scalability (group size)	Medium*1 ($\leq 8k$)	Huge (∞)	Poor (375–92)*4	Medium (4k–2k)*1,4
Processing time	$\mathcal{O}(M)$	$\mathcal{O}(G)$–$\mathcal{O}(SG)$*3	$\mathcal{O}(D)$	$\mathcal{O}(E)$
Packet overhead	Huge ($\mathcal{O}(M)$)	Low ($\mathcal{O}(1)$)	Very High ($\mathcal{O}(D)$)	High ($\mathcal{O}(E)$)
Cost (joining)	$\mathcal{O}(D \log M)$	$\mathcal{O}(RD)$	$\mathcal{O}(k)$	$\mathcal{O}(ED)$
Cost (leaving)	$\mathcal{O}(D \log M)$	$\mathcal{O}(R \log D)$	$\mathcal{O}(\log D)$	$\mathcal{O}(E \log D)$
Packet type	Proactive	Reactive	Proactive	Proactive+Reactive
Security	High	Medium*1,3	Low*1,3	Low*1,3
Convergence time	Low	High	Low	Medium*1
Optimal delivery	Yes	Uncertain*3	Yes	Yes*3
Asymmetric Problem	No	Yes	No	No*3
Multipath support	Yes	No	Yes	No*3
Header modification	No	No	Yes	Yes

*G: # of (active) groups; S: # of sources; D: # of receivers; I: # of interfaces, R: # of on-tree switches, E: # of edge switches, M: length of MCID. *1. Depend on DCN architecture. *2. Depend on memory usage on switch. *3. Depend on protocol. *4. Depend on IP version.*



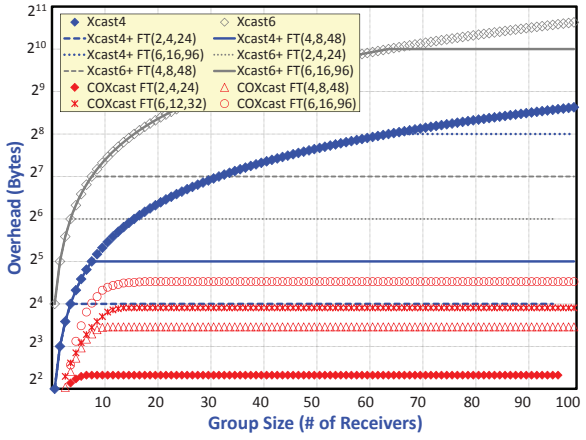
(a) (b)

Fig. 2. Required unicast overheads vs. (a) *FT* and (b) *BCube* network sizes.

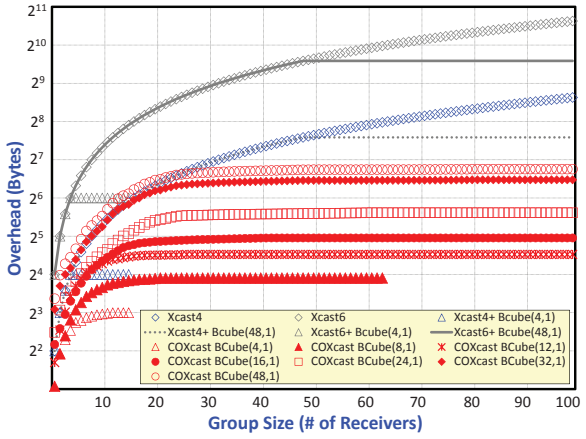
FT(4,8,48) and *FT*(6,16,96), respectively. Similar to previous results, many receivers can lead to longer MCIDs in COXcast. A longer MCID causes a larger overhead, so reduced effectiveness of the payload is expected. The difference is that the overhead of Xcast is only affected by the group size, whereas the overhead of COXcast is affected not only by the networks size, but also by the group size. Thus the Xcast's overhead is constant with increasing numbers of destinations in any size of DCN. The simulation results are depicted in Figures 3(a) and (b) which display results for Xcast and Xcast+ respectively. We find that the overhead of COXcast increases faster than Xcast as the group size increases; but with the group size continuously increasing (density), the increasing overhead slows down until the overhead is more than that of Xcast. Then overhead remains stable after the multicast tree spans all switches in the reference network. As regards *FT*(6,16,96), a group consisting of a 96 hosts is more efficient than that in Xcast. Another interesting observation from Figure 3(b) is that the overhead of COXcast is higher than Xcast4+'s for the network size *FT*(4,8,48). Without considering this factor, COXcast was superior for most remaining network sizes.

Although a practical DCN currently rarely exceeds 500 switches, we cannot rule out this possibility that to face higher scalability requirements in the near future. Compared to Xcast and Xcast+, the simulation results indicate that COXcast offers remarkable performance in scalability while simplifying the deployment and management of multicast service for medium-scale or even large-scale groups (depending on required payload size for specific applications) in large-scale DCNs. In a *FT*(8,16,96) network, the maximum number of receivers can reach 1,535, whereas the COXcast header size merely reaches 211 bytes with an effective payload of 1,300+ on standard MTU. Assuming that the network scale has sustainable growth of DCNs, COXcast still can accommodate more receivers in a group with acceptable header overhead, and almost unlimited number of groups.

The simulation results are also summarized in Table II which shows the maximum overheads in COXcast and its competitors for various network sizes from *FT*(2,4,16) to *BCube*(48,1) respectively. We have assumed that each situation has the maximum receivers applied (e.g., broadcast), and that COXcast has equal overhead for both IPv4 and IPv6. As shown by the results, the COXcast reduce the overhead in the best case by 99.77%, 99.92%, 84.38% and 96.09% compared to Xcast4, Xcast6, Xcast4+, and Xcast6+ respectively. In the worst case, the COXcast reduce the overhead by 70.1%, 92.52%, and 53.19% compared to Xcast4, Xcast6, and Xcast6+, respectively; while it increases the overhead by 87.24% in the worst case (*BCube*(12,3)) of Xcast4+. Based on the MTU=1.5K bytes, the Xcast cannot operate efficiently in all network sizes unless their receivers are sparse, the COXcast can still accommodate a sufficient payload size for all network sizes and any group size. The quantity is much larger than the theoretical maximum number of receivers, 374 and 92, in Xcast4 and Xcast6 respectively, even with an empty payload. At its least efficient situation, COXcast will carry for more payloads than either Xcast or Xcast+ are capable of.



(a)



(b)

Fig. 3. Required multicast overheads vs. group sizes in (a) *FT* and (b) *BCube* networks.

TABLE II
MAXIMUM OVERHEADS VS. NETWORK SIZES

Network Types	COXcast	Xcast4	Xcast6	Xcast4+	Xcast6+
FT(2,4,16)	10	12	1,008	16	64
FT(2,4,24)	14	380	1,520	16	64
FT(2,4,32)	18	508	2,032	16	64
FT(4,8,16)	22	508	2,032	32	128
FT(4,8,24)	30	764	3,056	32	128
FT(4,8,32)	38	1,020	4,080	32	128
FT(4,8,48)	54	1,532	6,128	32	128
FT(4,8,96)	102	3,068	12,272	32	128
FT(6,12,16)	36	764	3,056	48	192
FT(6,12,24)	48	1,148	4,592	48	192
FT(6,12,32)	60	1,532	6,128	48	192
FT(6,12,48)	84	2,300	9,200	48	192
FT(6,12,96)	156	4,604	18,416	48	192
FT(6,16,16)	47	1,020	4,080	64	256
FT(6,16,24)	63	1,532	6,128	64	256
FT(6,16,32)	79	2,044	8,176	64	256
FT(6,16,48)	111	3,068	12,272	64	256
FT(6,16,96)	207	6,140	24,560	64	256
FT(8,16,16)	51	1,020	4,080	64	256
FT(8,16,24)	67	1,532	6,128	64	256
FT(8,16,32)	83	2,044	8,176	64	256
FT(8,16,48)	115	3,068	12,272	64	256
FT(8,16,96)	211	6,140	24,560	64	256
BCube(4,1)	11	60	240	16	64
BCube(4,2)	126	252	1,008	64	256
BCube(4,3)	2176	1,020	4,080	16	64
BCube(8,1)	42	252	1,008	32	128
BCube(8,2)	1,752	2,044	8,176	256	1,024
BCube(8,3)	132,864	16,380	65,520	2,048	8,192
BCube(12,1)	93	572	2,288	48	192
BCube(12,2)	8478	6,908	27,632	576	2,304
BCube(12,3)	1,501,632	82,940	331,760	6,912	27,648
BCube(16,1)	164	1,020	4,080	64	256
BCube(16,2)	26,208	16,380	65,520	1,024	4,096
BCube(24,1)	366	2,300	9,200	96	384
BCube(24,2)	129,816	55,292	221,168	2,304	9,216
BCube(32,1)	648	4,092	16,368	128	512
BCube(32,2)	405,888	131,068	524,272	4,096	16,384
BCube(48,1)	1,452	9,212	36,848	192	768

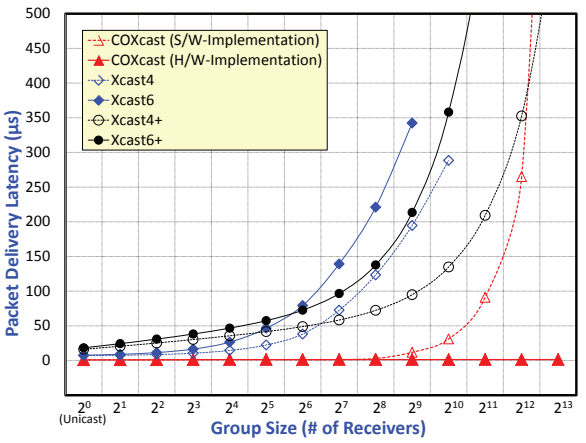


Fig. 4. Packet forwarding latencies vs. group sizes in *FT*(6,16,96) networks.

B. Multicast Packet Forwarding Latency

In this subsection, we compare packet forwarding performance of the COXcast with existing schemes. The simulation scenario is as follows: We randomly select several hosts in a *FT*(6,16,96) network. Each host serves over 2^{10} multicast groups in the meantime, and each group size=1$\sim 2^{13}$ increasingly. In addition, we assume that each random memory access in each switch requires 50ns, and the computing power of each switch to be fixed at 1 MIPS 32-bit microprocessor. The link capacity is 1Gbps at both edge level and server farm, and 10Gbps at core level, and each link has sufficient bandwidth and non-blocking switch capacity to accommodate the traffic need in the reference network. For the following simulations, we have considered the payload of each multicast packet is a constant 20 bytes in length.

Figure 4 presents the average ene-to-end packet forwarding latencies versus group sizes (number of receivers) in the reference DCN. We compare COXcast with various existing works, as described above, where the number of groups is fixed at 2^{13}. The curves show that S/W-implemented COXcast is much efficient than all existing schemes in the packet processing latency with no more than 2^9 receivers, and H/W-implemented COXcast has an advantage over any existing schemes. Now we can see that the Xcast protocols are affected by the group size, and so the massive processing cost leads to poorer performance than others when the group size is large. On the contrary, the curve for the core switches of Xcast+ are almost unaffected by increasing the group size, and it is obvious that the latency of edge switches of Xcast+ are increased with the group size. This is because the Xcast+ edge switches are responsible for the layer-2 table lookup of end receivers. Our scheme consistently obtains shorter average packet processing latency than other stateless approaches. The results fully demonstrate that our scheme is effectively reduces the forwarding latency in the DCNs.

V. Conclusion

In this paper, we present a novel self-routing unified unicast and multicast routing architecture in DCNs, called COXcast, to overcome the scalability problems in terms of stateless-based multicast. The proposed scheme entirely eliminates multicast forwarding states in the intermediate switches/routers by explicitly encoding the ciphered list of the forwarding set in the packets instead of using a multicast group address. We present the advantages and disadvantages, as well as the comparison with various existing work by main characteristics. The simulation results indicate that our proposed scheme offers remarkable performance in reducing the overhead and delivery latency, and improving the scalability of explicit multicast compared with the Xcast family scheme in various mainstream DCN topologies. Thus COXcast will effectively support the growing demands of real-time, large-scale, few-to-few multicast applications, and guarantees such emerging applications against all ineffective information which delay in arrival, hence the bandwidth are saved. The COXcast simplifies deploying and managing inter-domain and intra-domain multicast and enables a novel category of multicast routing scheme for large-scale DCNs. In addition, dual applications working with both IPv4 and IPv6 are recommended in future DCNs. But the development of IP version-independent technology and applications is required. COXcast avoids IP dependencies issue as a better solution that provides an IP version-independent architecture to applications and that hides all dependencies. Of particular note, COXcast is a natural multipath-enabled technique of using multiple alternative forwarding paths through a DCN, which can yield a variety of benefits such as fault tolerance, load balancing, bandwidth aggregation, multipath and improved security.

References

[1] A. Benslimane, *Multimedia Multicast on the Internet*. London, UK: ISTE, Wiley, 2010.

[2] J. Moy, "Multicast Extensions to OSPF," RFC 1584, Internet Engineering Task Force, Mar. 1994.

[3] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "An architecture for wide-area multicast routing," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 126–135, Oct. 1994.

[4] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (cbt)," *SIGCOMM Comput. Commun. Rev.*, vol. 23, no. 4, pp. 85–95, Oct. 1993.

[5] D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol," RFC 1075, Internet Engineering Task Force, Nov. 1988.

[6] R. Boivie, N. Feldman, Y. Imai, W. Livens, and D. Ooms, "Explicit Multicast (Xcast) Concepts and Options," RFC 5058, Internet Engineering Task Force, Nov. 2007.

[7] L. H. Sahasrabuddhe and B. Mukherjee, "Multicast routing algorithms and protocols: a tutorial," *Netwrk. Mag. of Global Internetwkg.*, vol. 14, no. 1, pp. 90–102, Jan. 2000.

[8] T. Braun and L. Liu, "Multicast for small conferences," in *Proc. 6th IEEE Symposium on Computers and Communications (ISCC 2001)*, Hammamet, Tunesia, Jul. 2001, pp. 145–150.

[9] R. H. Boivie, N. K. Feldman, and C. Metz, "Small group multicast: A new solution for multicasting on the internet," *IEEE Internet Computing*, vol. 4, no. 3, pp. 75–79, May/Jul. 2000.

[10] A. Bianco, P. Giaccone, E. M. Giraudo, F. Neri, and E. Schiattarella, "Multicast support for a storage area network switch," in *GLOBECOM*, Nov. 2006, pp. 1–6.

[11] C. Hu, X. Zhang, K. Zheng, Y. Chen, and A. V. Vasilakos, "Survey on routing in data centers: insights and future directions," *IEEE Network*, vol. 25, no. 4, pp. 6–10, Jun./Aug. 2011.

[12] K.-S. P. Myung-Ki Shin, Yong-Jin Kim and S.-H. Kim, "Explicit multicast extension (xcast+) for efficient multicast packet delivery," *ETRI Journal*, vol. 23, no. 4, pp. 202–204, Dec. 2001.

[13] A. Boudani, A. Guitton, and B. Cousin, "Gxcast: Generalized explicit multicast routing protocol," in *Proc. 9th IEEE Symposium on Computers and Communications (ISCC 2004)*, Jun. 2004, pp. 1012–1017.

[14] I. Stoica, T. S. E. Ng, and H. Zhang, "Reunite: A recursive unicast approach to multicast," in *Proc. 19th IEEE INFOCOM*, Apr. 2000, pp. 1644–1653.

[15] L. H. M. K. Costa, S. Fdida, and O. Duarte, "Hop by hop multicast routing protocol," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 249–259, Aug. 2001.

[16] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008.

[17] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 63–74, Aug. 2009.

[18] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 39–50, Aug. 2009.

[19] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," *Commun. ACM*, vol. 54, no. 3, pp. 95–104, Mar. 2011.

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

[21] D. Newman, "10 gig access switches: Not just packet-pushers anymore," *Netw. World*, vol. 25, no. 12, pp. 34–39, Mar. 2008.

[22] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[23] H. Holbrook and B. Cain, "Source-Specific Multicast for IP," RFC 4607, Internet Engineering Task Force, Aug. 2006.

[24] H. Wessing, H. Christiansen, T. Fjelde, and L. Dittmann, "Novel scheme for packet forwarding without header modifications in optical networks," *J. Lightwave Technol.*, vol. 20, no. 8, pp. 1277–1283, Aug. 2002.

[25] J. Postel, "Internet Protocol," RFC 791, Internet Engineering Task Force, Sep. 1981.

[26] J. Abley, P. Savola, and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6," RFC 5095, Internet Engineering Task Force, Dec. 2007.

Wen-Kang Jia (S'09–M'11) received his Ph.D. degree from the Department of Computer Science, National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2011. Before returned to school, he had been a senior engineer and manager since 1991 in various networking areas including ICT Manufacturer, Network Integrator, and Telecomm Service Provider. His recent research interests are the OSI layer-2∼5 such as TCP/IP protocols design, mobile management, error resilience coding, multimedia communications, NAT traversal, routing and switching, multicasting and broadcasting, teletraffic engineering, IP-optical convergence networks, P2P overlay networks, and wireless networks. He has published over 15 journal articles, over 30 international conference papers, 3 book chapters, and 3 patents. His work has been cited more than 300 times.