Reliable Multicast in Data Center Networks

Dan Li, Mingwei Xu, Ying Liu, Xia Xie, Yong Cui, Jingyi Wang, and Guihai Chen

Abstract—Multicast benefits data center group communication in both saving network traffic and improving application throughput. Reliable packet delivery is required in data center multicast for data-intensive computations. However, existing reliable multicast solutions for the Internet are not suitable for the data center environment, especially with regard to keeping multicast throughput from degrading upon packet loss, which is norm instead of exception in data centers. We present RDCM, a novel reliable multicast protocol for data center network. The key idea of RDCM is to minimize the impact of packet loss on the multicast throughput, by leveraging the rich link resource in data centers. A multicast-tree-aware backup overlay is explicitly built on group members for peer-to-peer packet repair. The backup overlay is organized in such a way that it causes little individual repair burden, control overhead, as well as overall repair traffic. RDCM also realizes a window-based congestion control to adapt its sending rate to the traffic status in the network. Simulation results in typical data center networks show that RDCM can achieve higher application throughput and less traffic footprint than other representative reliable multicast protocols. We have implemented RDCM as a user-level library on Windows platform. The experiments on our test bed show that RDCM handles packet loss without obvious throughput degradation during high-speed data transmission, gracefully respond to link failure and receiver failure, and causes less than 10% CPU overhead to data center servers.

Index Terms—Data center networks, reliable multicast, backup overlay

1 Introduction

Cloud computing is emerging as an attractive Internet service model [1], [2]. Giant data centers, with tens of thousands of, or even hundreds of thousands of servers, are built by cloud providers to offer various kinds of cloud services to tenants, leveraging the rapid growth of the Internet bandwidth. Group communication widely exists in data centers, from front-end delay-sensitive cloud applications, to back-end bandwidth-hungry infrastructural computations. Application examples include directing search queries to a set of indexing servers [3], distributing executable binaries to a group of servers participating cooperative computations such as MapReduce [4]–[6], upgrading OS and software on data center servers, replicating file chunks in distributed file systems [6], [7], etc.

Multicast benefits data center group communications in at least two aspects. By saving network traffic, it can increase the throughput of bandwidth-hungry computations such as Map-Reduce [4] and GFS [7]. By releasing the sender from sending multiple copies of packets to different receivers, it can also reduce the task finish time of delay-sensitive applications such as on-line query index lookup. However, existing multicast protocols built in data center switches/servers are originally designed for the Internet. Before the wide deployment of multicast in data centers, we need to carefully investigate whether these Internet oriented multicast protocols can well match the data center environment.

In this paper, we study a critical requirement of data center multicast, i.e., reliable data transmission. Transmission reliability is important because it determines the computation correctness of upper-layer applications, and accordingly the SLA (Service Level Agreement) of cloud services. Packet loss in data center multicast trees are supposed to be norm instead of exception for several reasons. First, current data centers are built by commodity servers/switches for economic and scalability reasons [15], [14], [8], [9], [11]. Packets can get lost from node/link failure in the fragile multicast trees. Second, traffic is quite bursty and unpredictable in data center networks [12], [13]. It is found that during traffic burst, the packet loss ratio can be as high as 12% [13], which severely hurts application performance. Our measurement results also show that the packet loss ratio in data centers can be up to 0.8% when traffic rate is high. Since it is difficult to reactively schedule multicast flows to low-utilized links, traffic congestion and consequent packet loss can occur at anyplace in the multicast tree. Third, the server population in data center is usually very large. The larger size the multicast tree is, the higher probability a packet gets lost during transmission.

Previous reliable multicast protocols for the Internet can be generally divided into two categories, i.e., network-device assisted and end-host based. The former category, represented by PGM [22] and ARM [23], requires assistance from network devices to achieve reliable data delivery. But the technical trend of data center design is to use low-end commodity switches, which are not supposed to bear much intelligence, for server interconnection [11]. Hence, the latter category of solutions, which put all the reliable multicast intelligence onto end hosts, seem to be more suitable for data center networks. The typical proposals include SRM [18], RMTP [19], TMTP [20], LBRM [21], etc. Though these protocols have been proposed for long, the managed environment

- *D. Li, M. Xu, Y. Liu, Y. Cui, and J. Wang are with Tsinghua University, 100084 Beijing, China. E-mail: tolidan@tsinghua.edu.cn.*
- *X. Xie is with Huazhong University of Science and Technology, 430074 Wuhan, Hubei, China.*
- *G. Chen is with Shanghai Jiaotong University, 200240 Shanghai, China.*

Manuscript received 06 Mar. 2012; revised 30 May 2012; accepted 03 Apr. 2013. Date of publication 16 Apr. 2013; date of current version 15 July 2014. Recommended for acceptance by K. Li.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2013.91

and rich link resource in data center networks offer unique opportunities for us to revisit the design space. We pay specific attention to the multicast throughput during packet repair, which is important for the task finish time of many cloud applcations.

The reliable data center multicast protocol we design is called RDCM, an end-host based protocol. Observing that current data centers are built with high link density, for each multicast group we explicitly construct a multicast-tree-aware backup overlay upon group members for packet repair. The construction and maintenance of the multicast tree and backup overlay is completed by a centralized multicast manager, which leverages the controlled environment of data centers. In case of packet loss, repair packets are transmitted in a peer-to-peer way on the backup overlay. Given multiple equal-cost paths between two servers, packet repair path in the backup overlay has high probability to be disjoint with the multicast tree. The advantage of this approach is multi-faceted. First, repair isolation is completely achieved by packet repair riding on unicast. No receiver will get the same packet twice, helping achieve high multicast throughput in high-speed multicast sessions. Second, when node/link failure happens, affected receivers can still get the repair packets via the backup overlay. Third, given packet loss from traffic congestion in the multicast tree, backup-overlay based packet repair can alleviate the congestion in hot spots and thus enhance the multicast throughput.

Building the backup overlay is one of the core technical challenges in RDCM. We carefully organize the backup overlay in such a way that each receiver is responsible for repairing packets to at most *two* other receivers, no matter how large the group size is. The multicast sender retransmits a packet only when all receivers lose it. The control-state exchange and overall repair traffic are also limited. Besides, the backup overlay can help detect and identify highly congested or failed links, assisting in reconstruction of the multicast tree and backup overlay.

RDCM realizes congestion control, by accommodating the sending rate of the multicast sender to the traffic status in the network. The traffic sending rate is controlled to be no higher than the receiving rate at the lowest receiver. The congestion control mechanism is also TCP-friendly, in the sense that the lowest receiver achieves the throughput as if a TCP unicast connection were running between itself and the sender. We choose a window based congestion control in RDCM. Each individual receiver maintains a congestion window, the size of which is updated using the AIMD (additive increase multiplicative decrease) algorithm.

We carry out simulations to compare the application throughput and traffic footprint in RDCM with other representative reliable multicast protocols (SRM, LBRM, RMTP) in BCube network. In all the packet loss ratios we test, RDCM achieves higher application throughput than all the other schemes, and exposes the least traffic footprint. Hence, RDCM can not only accelerate the computation progress of data center applications, but also reduce the usage of network resource. We also evaluate the congestion control mechanism of RDCM and the impact of the time to detect packet loss in simulations.

We have implemented RDCM as a user-level library on Windows platform for legacy IP multicast applications. The experiments in a 16-server test bed show that RDCM brings less than 10% CPU overhead to data center servers. Packet loss can be gracefully handled during high-speed data transmission without obvious multicast throughput degradation. In addition, multicast tree can be seamlessly adjusted upon link failure as well as receiver failure.

The rest of this paper is organized as follows. Section 2 discusses the background knowledge and design rationale. Section 3 presents the design of RDCM. Section 4 conducts simulations to study the performance of RDCM. Section 5 describes RDCM implementation and the experimental results. Section 6 presents the related work. Section 7 concludes the paper.

2 Background and Design Rationale

In this section, we discuss the background and design rationale of reliable multicast in data center networks.

2.1 Data Center Multicast

One-to-many group communication is common in modern data centers running cloud based applications. Multicast is the natural technology to benefit this kind of communication pattern, for the purposes of both saving network bandwidth and reduce the load on the sender. Services such as Facebook and Twitter are essentially supported by multicast-centric architectures [17]. For web search services, the incoming user query is directed to a set of indexing servers to look up the matching documents [3]. Multicast can help accelerate the directing process and reduce the response time. Distributed file system is widely used in data centers, such as GFS [7] in Google, HDFS [6] in Hadoop, and COSMOS in Microsoft. Files are divided into many fix-sized chunks, say, 64 MB or 100 MB. Each chunk is replicated to several copies and stored in servers located in different racks to improve the reliability. Chunk replication is usually bandwidth hungry, and multicast-based replication can save the inter-rack bandwidth. In map-reduce like cooperative computations [4]–[6], the executive binary is delivered to the servers participating the computation task before execution. Multicast can also speed up the binary delivery and reduce the task finish time.

Though multicast is supported by most network devices (routers, switches) and end hosts, it is not widely deployed in the Internet due to many technological causes, such as the pricing model, multicast congestion control, security concerns. For the same reason, modern data centers rarely enable multicast protocols. To avoid the traffic redundancy in group communications, one possible solution is to divide the network into VLANs, and use broadcast within each VLAN. For example, for map-reduce like jobs, we can easily configure the workers for the same job in one VLAN, and perform broadcast within the VLAN to deliver the executable binary. However, there are two problems for this approach. First, the VLAN tag space is quite limited. There can be at most 4k VLANs since the VLAN tag ID is 12 bits [35]. But the potential number of multicast groups in data center can be very large, especially considering the file chunk replication groups. Second, dynamical group member join/leave is common, e.g., creating new workers/VMs to handle failure in map-reduce jobs. It has a high cost to dynamically reconfigure the VLANs in

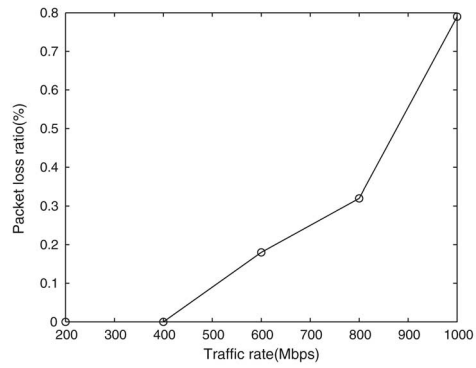


Fig. 1. Packet loss ratio against traffic rate in our test bed.

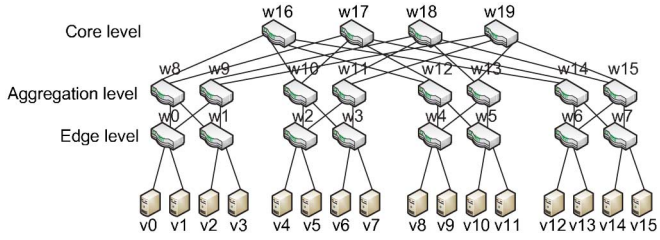


Fig. 2. A Fat-Tree architecture connecting 16 servers. It has three levels of switches, each with 4 1G ports.

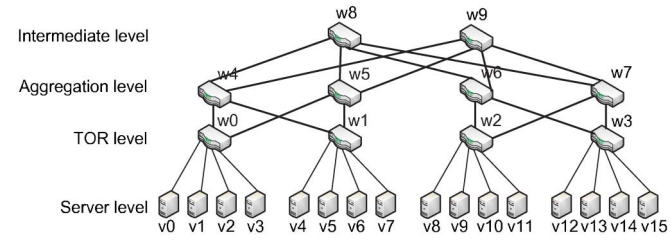


Fig. 3. A VL2 architecture connecting 16 servers. The TOR-level switches have 4 1G ports and 2 10G ports. The aggregation-level and intermediate-level switches are all built with 4 10G ports.

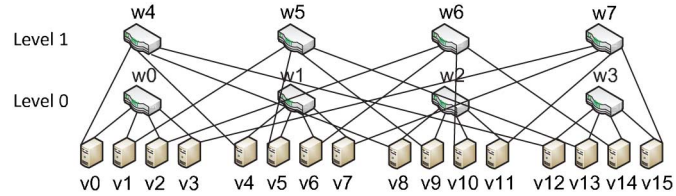


Fig. 4. A BCube architecture connecting 16 servers. It is a BCube_1 constructed upon 4 BCube_0s and another level of 4 switches. Each server has 2 1G ports and each switch has 4 1G ports.

response to group membership change. However, multicast does not have the limitations above. The IP multicast address space can support hundreds of millions of groups, and can naturally embrace group member dynamics.

Therefore, we argue that IP multicast (or more generally, network-level multicast) is the preferred choice to support data center group communication, especially considering that we can leverage the managed environment of data centers to overcome the open problems for IP multicast in the Internet. However, current multicast protocols implemented at switches and servers are primarily Internet oriented. Before the wide deployment of multicast in data center networks, we need to carefully investigate whether these multicast protocols can well embrace the data center environment. In this paper, we focus on a specific problem for data center multicast, i.e., reliable data delivery.

Reliable packet delivery is important for data center multicast, because packet loss ratio in data centers can be high when traffic rate is high. Fig. 1 shows the packet loss ratio in a 1Gbps link in our small data center test bed composed of 20 servers. We find that when the traffic rate is lower than 400Mbps, there is almost no packet loss. However, the packet loss ratio grows significantly with higher traffic rate. When the traffic rate reaches the full link capacity, the packet loss ratio can be as high as 0.8%.

2.2 Data Center Network Architecture

In current practice, data-center servers are connected by a tree hierarchy of Ethernet switches, with commodity ones at the first level and increasingly larger and more expensive ones at higher levels. It is well known that this kind of tree structure suffers from many problems [14], [15]. The top-level switches are the bandwidth bottleneck, and high-end high-speed switches have to be used. Moreover, a high-level switch shows as a single-point failure spot for its subtree branch. Using redundant switches does not fundamentally solve the problem but incurs even higher cost.

To overcome the limitations of tree structure, recently many new data center architectures are proposed [14], [8], [9]. A consistent theme in these new architectures is that several levels of low-end commodity switches are used to interconnect a massive number of servers. The major difference among the proposals lies in the way how switches are interconnected and how servers are connected to switches. Figs. 2–4 show the examples of how to connect 16 servers in Fat-Tree, VL2 and BCube respectively. In Fat-Tree [14], each server uses one 1G NIC port to connect an edge-level 1G switch, and adjacent levels of 1G switches are interconnected via a Fat Tree structure. VL2 takes a similar fashion [8]. Every server uses a 1G link to connect a ToR-level switch. Each ToR-level switch uses two 10G uplinks to connect two aggregation-level switches respectively. Aggregation-level switches and intermediate-level switches are connected as a complete bipartite graph using 10G links. In BCube [9], each server uses $k+1$ ports to connect $k+1$ switches from different levels. Any two switches are not directly connected.

Note that the link density in these modern data center networks is very high. Hence, there are many equal-cost paths between any two servers. For instance, in a Fat-Tree structure composed of k-port switches, there are $\frac{k^2}{4}$ core-level switches, which equals to the number of paths between any two servers. In VL2, if k-port 10G switches are used for interconnection, the number of equal-cost paths between any two servers is $4k$. In a BCube(n,k) network, where n is the number of switch ports and $k+1$ is the number of server ports, the number of equal-cost shortest paths between two servers is typically $(k+1)!$. If we relax the path length requirement, the candidate paths between BCube servers is even more. The rich link resource in data center networks exposes both new challenges and opportunities for data center protocol design [10].

2.3 Design Rationale

Due to the essential unreliability of UDP-based multicast transmission, reliable mechanism should be introduced to guarantee the successful packet delivery to multicast receivers. We target at single-source multicast, in which a single sender distributes data to a number of interested receivers. Note that the most bandwidth-efficient data delivery structure for IP multicast is the tree structure, which utilizes the least number of links for a multicast group. Considering the rich link resource in typical data center networks, it makes sense to build multiple trees for a group to accelerate the data distribution process [9], [24]. Though, tree is still the basic structure for managing data delivery.

There are several challenges for designing reliable multicast in data center environment.

Fragile Multicast Trees: In considerations of economical cost and scalability, current data centers are built upon a large number of commodity switches and servers. Failure is norm instead of exception in such networks [8], and the multicast tree is quite fragile. Any node/link failure in the multicast tree can pause packet delivery to downstream receivers. Reliable multicast requires gracefully handling node/link failure during packet transmission.

Traffic Bursty in Data Centers: It has been shown that traffic is quite bursty and unpredictable in data center networks [12]. When the group size is large, traffic congestion can occur anywhere in the multicast tree, resulting in frequent packet loss. When transmitting the repair packets, the multicast throughput will degrade significantly if the repair packets compete for the network bandwidth with regular packets in the multicast session.

Design Intelligence: The low-end commodity switches used in current data centers usually contain quite limited routing states, small buffer space as well as low programmability. These switches are not supposed to bear much multicast intelligence, except the basic multicast packet forwarding. Hence, network-device assisted reliable multicast solutions are not suitable for data center environment.

To address the challenges above, we design RDCM, a novel reliable multicast protocol for data center networks, by leveraging the managed environment of data centers, the rich link resource in data center networks, as well as the topological characteristics of modern data center networks. RDCM makes the following design choices.

First, RDCM uses a central controller to build the multicast tree, since data center is usually a managed environment controlled by a single organization. This kind of centralized controller is widely adopted in modern data center design. For instance, in Fat-Tree [14], a fabric manager is responsible for managing the network fabric. In VL2 [8], a number of directory servers are used to map the AA-LA relationship. The emerging OpenFlow [16] framework also uses a controller for routing rule decision and distribution. In RDCM, we call the controller *multicast manager*. All the group join/leave requests are redirected to the multicast manager. The multicast manager calculates the multicast tree based on the network topology and group membership, and configures the forwarding states on switches/servers.

Second, RDCM takes a peer-driven approach to packet repair, leveraging the rich link resource in data center networks. If the source retransmits the repair packets along the multicast tree, the repair packets will compete for the link bandwidth with normal packets in the multicast session. RDCM, instead, repairs lost packets among receivers by P2P unicast. Given the rich link resource and multiple equal-cost paths between any two servers in data center networks, the packet repair paths has high probability to be disjoint with the multicast tree, which we will further elaborate in Section 3.

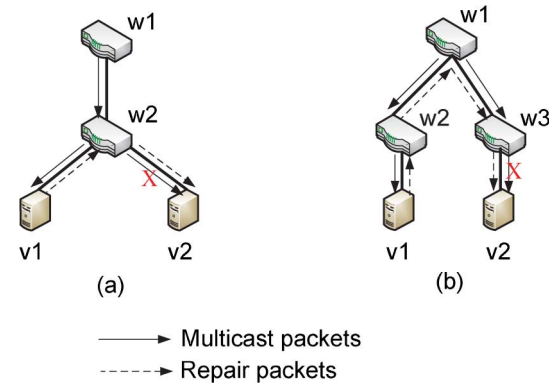


Fig. 5. Examples to show the packet transmission path along the multicast tree and the packet repair path. In (a), the packet is transmitted only once in each link. In (b), the packet is transmitted twice in the link $w1 \rightarrow w3$. However, it can be avoided if the unicast repair path can bypass the multicast tree.

Third, RDCM explicitly constructs a multicast-tree-aware repair structure to improve the packet repair efficiency, which we call *backup overlay*. Peer-driven unicast has also been used for packet repair in some Internet-oriented reliable multicast schemes, such as Bimodal Multicast [28], Lightweight Probabilistic Broadcast [29] and Ricochet [30]. However, the managed environment of data centers and the regular topology provide the unique opportunity for building a topology-aware overlay among receivers for packet repair. Compared with gossip-based packet repair schemes, explicitly constructing the backup overlay can help achieve repair isolation, control the individual repair burden and reduce the overall repair traffic, all of which favor enhancing the multicast throughput. All the switches are oblivious to the backup overlay.

Note that RDCM basically designed for generic data center topologies. In what follows, we primarily use BCube as an example when presenting the design details, and carry on simulations as well as experiments. But they can be easily extended to other data center networks.

3 Design

In this section, we present the design of RDCM.

3.1 Design Overview

As presented in Section 2, RDCM puts all the design intelligence to data center servers and uses peer-driven unicast for packet repair. In order to not degrade the multicast application throughput, it is desired that no link transmits the same packet more than once. We take Fig. 5 as an example, which is part of a multicast tree. In Fig. 5(a), assume a packet is lost in the link $w2 \rightarrow v2$, and receiver $v1$ is responsible for repairing

this packet to $v2$. As shown in the figure, the packet is transmitted only once in each link (note that the links are bi-directional), and hence the application throughput will not degrade. Then let's check Fig. 5(b), in which the packet loss occurs in the link $w3 \rightarrow v2$, and receiver $v1$ is responsible for repairing this packet to $v2$. If the repair packet is still transmitted along the multicast tree, the link $w1 \rightarrow w3$ will transmit the packet twice, which affects the application throughput.

Fortunately, as we described in Section 2, modern data center networks tend to have rich link resource and multiple equal-cost paths between servers. Hence, the unicast packet repair path has high probability to bypass the multicast tree. For example, in a Fat-Tree network with 48-port switches, there are typically 576 paths between two servers. Hence, if we use unicast packet repair and randomly choose a core-level switch, the repair path has a probability of $575/576 = 99.8\%$ to bypass the core-level switch used in the multicast tree. The probability is also high to bypass the low-level switches in the multicast tree. In a typical BCube(n,k) network, between any two servers, there are $(k+1)!$ shortest paths and $k+1$ parallel shortest paths. Consequently, when using unicast for packet repair, with a probability higher than $\frac{k}{k+1}$ that the repair path bypasses the multicast tree.

Overall, there are several advantages to transmit the repair packets in a peer-to-peer unicast way in richly-connected data center networks. First, the repair burden on the multicast source is reduced. Second, when node/link failure occurs in the multicast tree, transmission pause can be avoided. Third, given link congestion in the multicast tree, the congested link will not be exacerbated by repair packets.

Existing Internet oriented reliable multicast protocols using peer-driven packet repair usually take a gossip way [28]–[30]. However, RDCM builds an explicit multicast-tree-aware backup overlay by leveraging the managed environment and the topological information of data center networks. Using the backup overlay, the packet repair responsibility among receivers is determined. Hence, we can both achieve repair isolation and avoid duplicate replication. RDCM gracefully handles link/switch failures by adjusting the multicast tree and the backup overlay. RDCM also realizes congestion control to accommodate the source sending rate to the traffic status. In the following subsections, we present the backup overlay construction, packet repair scheme, failure handling, as well as the congestion control mechanism in RDCM, respectively.

3.2 Backup Overlay

In RDCM, the multicast manager not only builds the multicast tree for a group, but also constructs a backup overlay on the multicast tree for reliable packet delivery. A backup overlay is composed of a number of *overlay rings*. Each branching node, i.e., the one with more than one children, in the tree, has a corresponding overlay ring. The overlay ring for a level-l (l starts from 0 at the lowest leaf level) branching node i, denoted as O_i, is called a *level-l overlay ring*. If node i has c children nodes in the tree, O_i is composed of c receivers. The c receivers are selected from each branch rooted from i. If j is one of i's children in the tree, the downstream receiver of j chosen to join O_i is called the *overlay proxy* for j, denoted as p_j. Hence, the length of an overlay ring is bounded by the number of switch ports in data center network. Within an overlay ring,

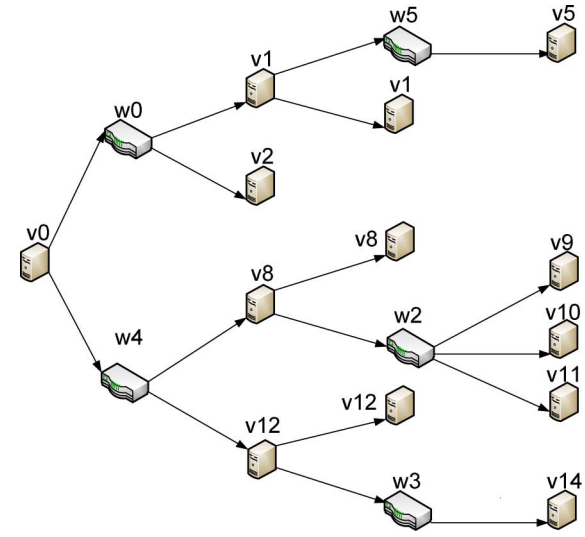


Fig. 6. A multicast tree in BCube network of Fig. 4. The sender is $v0$, and the receiver set is $\{v1, v2, v5, v8, v9, v10, v11, v12, v14\}$. The tree has 5 levels.

each receiver has an *overlay successor* and an *overlay predecessor*. In addition, i is called the *tree parent* for O_i, p_i is called the *overlay parent* for O_i, and the receivers in O_i are called the *overlay children* for p_i. Specifically, the sender is the overlay proxy for itself. It joins no overlay ring but it is both the tree parent and the overlay parent for the highest-level overlay ring in the backup overlay.

We take a multicast group from Fig. 4 as an example. The sender is $v0$, and the receiver set is $\{v1, v2, v5, v8, v9, v10, v11, v12, v14\}$. The multicast tree[1] is shown in Fig. 6. Note that for a BCube server which is both a receiver and a forwarder in the multicast tree, it is regarded as a switch connecting a child receiver of itself, such as $v1$, $v8$ and $v12$. The backup overlay upon the multicast tree is shown in Fig. 7. There are 7 overlay rings, each corresponding to a branching node in the multicast tree. For instance, the overlay ring O_{w4} is composed of two receivers, $v11$ and $v14$, which are the overlay proxies for $v8$ (switch) and $v12$ (switch) respectively. $w4$ is the tree parent for O_{w4}. $v8$ (receiver) is the overlay proxy for $w4$ and thus the overlay parent for O_{w4}. $v11$ and $v14$ in O_{w4} are both the overlay children of $v8$. Within O_{w4}, $v11$ and $v14$ are both the overlay predecessor and overlay successor for each other.

Fig. 8 illustrates the whole procedure to build the backup overlay for a multicast tree t. It takes a bottom-up way, i.e., from lowest-level branching tree nodes up to the sender. The overlay ring for a tree node i is constructed only after the overlay rings for all its children nodes are constructed. Then, if i has only one child, no overlay ring is built for it. Otherwise, for each of i's children, say j, its overlay proxy p_j is selected from its downstream receivers which has joined the least overlay rings; and all the overlay proxies for i's children form the overlay ring for i, i.e., O_i.

Still take Fig. 7 as the example. We first construct the overlay ring for the lowest-level branching node $w2$, which is composed of the three receivers, namely, $v9$, $v10$ and $v11$. It is called a level-1 overlay ring. Then there are three level-2

1. The algorithm to construct the multicast tree is beyond the scope of this paper.

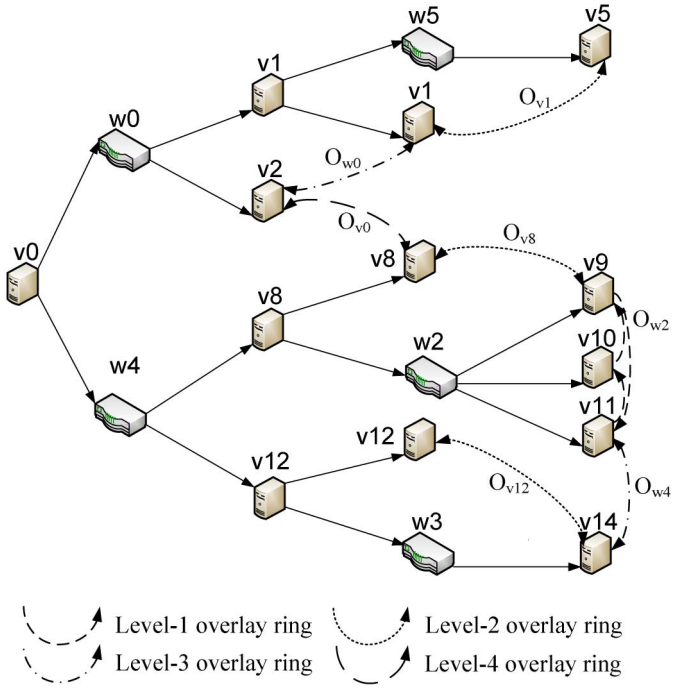


Fig. 7. The backup overlay upon the multicast tree in Fig. 6. It is composed of 4 levels of overlay rings. Level-1 overlay rings are for the lowest-level branching nodes, which only include O_{w2} in the figure. Level-2 overlay rings include O_{v1}, O_{v8} and O_{v12}. Level-3 overlay rings include O_{w0} and O_{w4}. Level-4 overlay ring only includes O_{v0}.

Denotations:
C_i - The set of i's children nodes in the multicast tree;
D_i - The set of receivers in the subtree rooted from i.

Algorithm:

```
1  void OverlayRingBuild(i){
2     if (|C_i| == 0) /*i is a leaf receiver*/
3        return;
4     for (j ∈ C_i)
5        OverlayRingBuild(j);
6     if (|C_i| == 1) /*i has only one child*/
7        return;
8     for (j ∈ C_i)
9        p_j = the receiver in D_j joining the least overlay rings;
10    Construct O_i on all p_j, j ∈ C_i;
11    return;
12 }/*OverlayRingBuild*/

1  void BackupOverlayBuild(TREE t){
2     s = root of t;
3     OverlayRingBuild(s);
4     return;
5  }/*BackupOverlayBuild*/
```

Fig. 8. Algorithm of constructing the backup overlay for a multicast tree t.

branching nodes, i.e., $v1$, $v8$ and $v12$. Take the overlay ring for $v8$, O_{v8} as an example. It has two branches. For one branch, we select the receiver $v8$. For the other branch, we select one downstream receiver which joins the least overlay rings (randomly choosing one among multiple candidates), $v9$. Hence, O_{v8} is formed by $v8$ and $v9$. Next, we construct the overlay rings for two level-3 branching nodes, i.e., $w0$ and $w4$. Take O_{w4} as an example. It has two branches, too. We select one receiver from each branch which joins the least overlay rings, namely, $v11$ and $v14$ in this example. Finally, for the source server $v0$, its overlay rings is composed of $v2$ and $v8$.

We make definitions on the overlay rings a receiver r joins.

Leaf Overlay Ring: If receiver r joins such an overlay ring for a tree node i, in which r is the only one receiver in the subtree rooted from one of i's children, the overlay ring is called r's *leaf overlay ring*.

Proxy Overlay Ring: If r joins an overlay ring which is not its leaf overlay ring, the overlay ring is called r's *proxy overlay ring*.

For instance, in Fig. 7, $v9$ joins two overlay rings. For the overlay ring O_{w2}, $v9$ is the only one receiver in the subtree rooted from $v9$ itself. Hence, O_{w2} is the leaf overlay ring for $v9$. The overlay ring O_{v8} is then the proxy overlay ring for $v9$ (but note that O_{v8} is the leaf overlay ring for $v8$). Similarly, for server $v1$, O_{v1} is its leaf overlay ring, and O_{w0} is its proxy overlay ring.

Based on the definition, we can easily find that a receiver has no overlay children in its leaf overlay ring. But if it joins a proxy overlay ring for tree node i, it must have overlay children, because in this case it is the overlay parent for the overlay ring for one of i's children in the tree. For server $v9$ in Fig. 7, it has no overlay children in the leaf overlay O_{w2}. But in the proxy overlay ring O_{v8}, $v9$ has three overlay children, namely, $v9$, $v10$ and $v11$.

Theorem 1. *In RDCM, every receiver joins exactly one leaf overlay ring if the group has more than one receivers, and joins at most one proxy overlay ring.*

Proof. Please refer to the Appendix, which can be found in the Computer Society Digital Library at https://doi.ieeecomputersociety.org/10.1109/TC.2013.91/. Additional mentions of supplemental material are available online at http://ieeexplore.ieee.org. □

3.3 Packet Repair Scheme

Packet repair in RDCM is carried on in a peer-to-peer way on the backup overlay.

Packet Acknowledgement: Each packet is assigned with a sequence number. A receiver r acknowledges a packet k in the following way. First, when r receives packet k, either from the multicast tree or from the backup overlay, it sends an ACK for k to both the overlay predecessor and the overlay parent in its leaf overlay ring. Second, when r receives the ACK for k from its overlay children for the first time, it sends an ACK for k to both the overlay predecessor and the overlay parent in its proxy overlay ring. All ACK packets are unicast. The hierarchy fashion of ACK is also used in RMTP [19].

Take Fig. 7 as an example, when $v8$ receives the packet k, it sends the ACK for k to $v9$ and $v11$, which are its overlay predecessor and overlay parent respectively in its leaf overlay ring, O_{v8}. When $v8$ receives the first ACK for packet k for a packet from one of its overlay children in overlay ring O_{w4} (either $v11$ or $v14$), it sends an ACK for k to $v2$ and $v0$, which are the overlay predecessor and the overlay parent respectively in its proxy overlay ring, O_{v0}.

In this way, in the leaf overlay ring a receiver joins, it receives ACK for each packet only from its overlay successor in the ring. If a receiver joins a proxy overlay ring, it also receives ACK for each packet from its overlay successor in the ring as well as its overlay children. The sender receives ACK

only from its overlay children. Note that the number of overlay children for a receiver is bounded by the number of ports in a switch, say, n, which is usually a small constant. As a result, a receiver/sender receives at most $n+2$ ACKs from other receivers, independent of the group size. ACK implosion is thus effectively avoided in RDCM. In practice, to further reduce ACK messages exchanged, a receiver can choose to send one ACK for multiple packets, instead of acknowledging each one in a separate packet.

Repair Window: Every receiver maintains a repair window for packets received from both the multicast tree and the backup overlay. Packets within the repair window are buffered. The upper bound of the repair window is the highest sequence number among the packets received. The lower bound of the repair window at a receiver r is moved up if all the following conditions are satisfied for the corresponding packet k. First, r has received k. Second, r has received ACK for k from its overlay successors of all the overlay rings it joins. Third, r has received an ACK for k from one overlay child if it joins a proxy overlay ring. Hence, each receiver only needs to wait for at most 3 ACKs before moving the window up and releasing the buffer.

Specifically, the sender also has a repair window. Since it is the overlay parent for the highest-level overlay ring, the repair window is moved up when receiving an ACK for the corresponding packet from any overlay child.

Packet Repair: In RDCM, repair packet is either unicast within an overlay ring or multicast by the sender. We first discuss packet repair within overlay rings. Each receiver is responsible for packet repair to its overlay successors in all the overlay rings it joins. If a receiver detects that its overlay successor has lost a packet (by timing out for the ACK), it immediately transmits the repair packet. From Theorem 1, every receiver is responsible for repairing packets to at most two other receivers, no matter how large the group size is. Hence, the repair burden for an individual receiver in RDCM is quite low. It not only balances the repair loads on receivers, but also disperses the repair traffic to the whole network.

When a packet gets lost in the incoming link of a level-l node i in the multicast tree, all the downstream receivers of i will lose the packet. In this case, p_i will receive the repair packet from the level-$(l+1)$ overlay ring it joins. After that, each downstream receiver of i transmits the repair packet to overlay successors after receiving it. In this way, the repair packet can be distributed to all the downstream receivers of i.

In the example of Fig. 7, assume a packet is lost in the incoming link to $w4$, then all downstream receivers of $w4$ will miss the packet. Then, $v8$, the overlay proxy for $w4$, will receive repair packet from $v2$ via the level-4 overlay ring it joins, O_{v0}. Note that $v2$ buffers the packet because it does not receive ACK for the packet from $v8$. The repair packet is unicast and is probable to bypass the incoming link to $w4$ in the multicast tree. After $v8$ receives the packet, it repairs it to $v9$ in the level-2 overlay ring of O_{v8}. Then $v9$ repairs to $v11$. Next, $v11$ sends the packet simultaneously to $v10$ and $v14$. Finally, $v14$ repairs to $v12$ in the level-2 overlay ring of O_{v12}, and all the downstream receivers of $w4$ receive the repair packet.

When the sender receives no ACK for a packet from its overlay children, all the receivers should lose this packet based on our design. Then the sender multicasts the repair packet to the whole tree. But this case should rarely occur.

Repair Traffic: In a typical backup overlay, most overlay rings are formed by leaf nodes in the multicast tree. Note that in the overlay rings formed by leaf nodes, the neighboring nodes are usually connected to the same switch and are only two hops away. For example, the overlay rings O_{v1}, O_{v2}, O_{v8}, O_{w2} and O_{12} in Fig. 7. Compared with the most traffic-saving packet repair method using multicast (or scoped multicast), our repair scheme only doubles the overall repair traffic for receivers experiencing packet loss. But we require no intelligence from switches to realize complete repair isolation. Besides, when severe packet loss occurs and a large number of packets are repaired on the backup overlay, RDCM can help adjust the multicast tree to a new one, as presented later.

Repair Latency: The repair latency in RDCM can be higher than other reliable multicast schemes. But it is limited for two reasons. First, when a receiver joining two overlay rings receives a repair packet, it simultaneously sends out the packet to both its overlay successors in the two overlay rings, and thus the repair packet is transmitted in parallel on the backup overlay. Second, dominant hop-by-hop repairs occur in the lowest-level overlay rings as we depend tree-based multicast, which crosses only two physical links. The time needed to detect packet loss also accounts for repair latency. But the number of hierarchies in the RDCM ACK is usually small because the diameter in modern data center networks is low. Note that RDCM is primarily designed for data-intensive applications in data centers, in which application throughput is more important than end-to-end latency. Hence, we argue that the repair latency in RDCM is acceptable for the applications it is intended to support.

3.4 Tree and Backup Overlay Adjustment

In RDCM, the multicast tree and the backup overlay are adjusted upon receiver dynamics, as well as node/link failure in the multicast tree.

Receiver Dynamics: When receiver dynamics occur, including receiver join, receiver leave or receiver failure, the multicast tree is reformed based on the updated membership. RDCM also adapts the backup overlay to the added/deleted members.

When a new receiver joins a multicast group, the multicast manager inserts the receiver to an appropriate position in the multicast tree. Then, the backup overlay is recalculated according to the new tree. When recalculating, RDCM minimizes the changes to the existing backup overlay. Typically, we only need to update the overlay ring for the branching tree node which adds a branch for the new receiver. The new receiver starts to get data from the multicast tree and the buffer of its overlay predecessors in the back overlay. Note that RDCM does not guarantee that newly joined receiver can receive the data from the first byte of the multicast session before its join.

When a receiver leaves the multicast session, the multicast manager deletes the leaving receiver from the multicast tree. The backup overlay is updated by changing the overlay ring for the branching tree node which deletes a branch for the leaving receiver. RDCM continues to transmit data to other receivers alive. For graceful receiver leave, the leave message is directed to the multicast manager, and it is easy to adjust the multicast tree and backup overlay. When receiver crash happens, RDCM tries to detect the failed ones. The detection

is easy to conduct by the backup overlay. Every receiver joins at least one overlay ring in the backup overlay and acknowledges the received packets to its overlay predecessor. Suppose $r2$ is the overlay successor of $r1$ in an overlay ring. When it timeouts for $r1$ to receive ACKs from $r2$, $r1$ transmits repair packets to $r2$. If $r1$ fails in the packet transmission via backup overlay, it is an indication that $r2$ has failed. Then, $r1$ sends a *receiver failure message* to the multicast manager, and the backup overlay will be accordingly adjusted.

Link Failure: When a multicast receiver gets all the packets from the backup overlay for a period of time, it is an indication that switch/link failure occurs in the multicast tree. Fortunately, RDCM can help detect and identify failed links in the multicast tree, and both the multicast tree and the backup overlay can be adjusted.

We let each receiver $r1$ monitor both its overlay successor $r2$. When $r1$ finds that $r2$ receives all the packets from the backup overlay (it is feasible since $r1$ is responsible for packet repair to $r2$), it sends a *link failure report* to the multicast manager. In this way, for a failed link in the multicast tree, only *one* receiver is responsible for sending the link failure report, no matter where the failed link lies. For instance in Fig. 7, when the link $w5 \rightarrow v5$ fails, $v1$ is responsible for sending the report.

The tree adjustment message from $r1$ contains the identity of $r2$. When the multicast manager receives the report, it infers the failed link as follows. At first it assumes there is a single link failure. The report implies that the tree parent for the overlay ring including both $r1$ and $r2$, say, tree node i, works well. Then, the *failed link set* is inferred to be composed of the links from node i down to the first branching node or $r2$ along the multicast tree. The failed link can be any one from this set, and not beyond. So the inference has *zero false negative*. We cannot further identify the exact failed link from the set. The multicast manager kicks all the links in this set off the data center topology, and accordingly updates the multicast tree as well as the backup overlay. The rich link resource in data center networks can tolerate the possible false positive of the failed link inference. We take Fig. 7 as an example. When the multicast manager receives a link failure report from $v1$ containing the identity of $v5$, it implies that $v1$ is working well and the failed link set is inferred as $\{v1 \rightarrow w5, w5 \rightarrow v5\}$. Then, the two links are deleted from the data center topology. The multicast tree and the backup overlay are recomputed and shown in Fig. 9.

If a link failure report is triggered by multiple link failures instead of a single one, we cannot guarantee that the adjusted tree above contains no failed links. However, in this case, another link failure report will be triggered and the tree can be further adjusted. Our backup overlay avoids transmission pause during the tree-switching period given that data center topology is not partitioned. The multicast manager maintains a global set to record the failed links inferred by link failure reports, and the information is shared among all the multicast groups. The failed links in the global set can be periodically activated to utilize the recovered ones. Of course, link failure information can also be obtained in out-of-band ways and integrated into RDCM to help adjust the multicast tree and the backup overlay.

Repair Window Management when Switching Overlay Rings: When the backup overlay is adjusted, the overlay rings a receiver joins can also change. If the receiver exits all the previous overlay rings and joins new overlay rings, there may exist a problem on repair window management. For example, a receiver $r1$ previously joins one overlay ring $O1$. Now it exits $O1$ and joins a new overlay ring $O2$. Assume the overlay predecessor of $r1$ in $O1$ is $r2$, while that in $O2$ is $r2'$. If $r1$ exits $O1$ immediately after joining $O2$, $r1$ may fail to receive its lost packets because $r2'$ has already released them in its repair window. To solve this problem, we let $r1$ still lie in $O1$ after joining $O2$, until when it can receive lost packets from $r2'$ in the new overlay ring. However, when a receiver crashes, its overlay successors may not be able to receive some lost packets. In this case, the affected receivers send specific requests to the multicast sender for these lost packets.

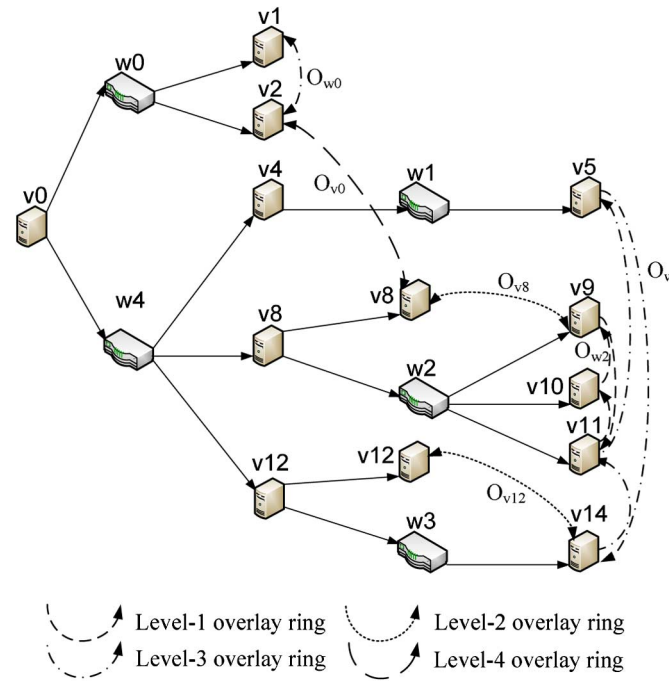


Fig. 9. The adjusted multicast tree and backup overlay from that of Fig. 7, if the failed link set is $\{v1 \rightarrow w5, w5 \rightarrow v5\}$.

3.5 Congestion Control

RDCM realizes congestion control, by accommodating the sending rate of the multicast sender to the traffic status. The traffic sending rate at the multicast sender should be no higher than the receiving rate at the lowest receiver. To be TCP-friendly, we let the lowest receiver achieves the throughput as if a TCP unicast connection were running between itself and the sender. For scalability consideration, the rate estimation algorithm is running at each individual receiver instead of at the sender. There are two basic congestion control approaches, namely, rate based and window based [31]. In rate based congestion control, each receiver calculates the receiving rate as the TCP throughput under the same packet loss ratio. However, this approach requires measuring the end-to-end delay between the sender and receiver. Accurately measuring the end-to-end delay is quite difficult in data center networks, since the delay is in the order of micro-seconds, which can be sensitive to various server/network conditions (e.g., the processing time in the sender/receiver). The measurement will also introduce significant burden on the sender.

Therefore, we prefer window based congestion control in RDCM. Each individual receiver maintains a congestion window, the size of which is updated using the BIC (Binary Increase Congestion Control) algorithm [37]. Packet loss is used as the signal for congestion. Note that most repair packets traverse data center links other than the multicast tree in RDCM, hence the repair packets have no/little contribution to the congestion on the multicast tree. As a result, for a receiver i whose congestion window size is w_i, if the maximum sequence number received is m_i, the highest expected sequence number within the next RTT is set as $h_i = m_i + w_i$. All the receivers report the highest expected sequence number to the multicast sender, and the sender sends out packets to accommodate the slowest one.

The congestion window on an RDCM receiver should be smaller than the repair window, because the repair window also needs to maintain the repair packets. Let's consider the case if repair packets pass the multicast tree. Then the repair packets should also be accounted in the congestion window. Hence, the highest expected sequence number in the next RTT on a receiver i is $h'_i = m_i + w_i - l_i$, where l_i is the number of lost packets below m_i. It is easy to get that there is $h_i > h'_i$. Compared with the packet repair schemes traversing the multicast tree, RDCM relaxes the constraints on the packets sent out by the multicast sender, and naturally enhances the multicast throughput.

Ideally, each receiver can immediately detect packet loss when it occurs, update the congestion window, and send to the sender. But in practice, the delay in detecting packet loss (depending on the length of timer) will result in some "lag" in responding to the congestion. The lag can cause further congestion and packet loss. But longer buffering queues in switches can mitigate the problem and help improve multicast throughput. We will evaluate the impact in Section 4.

4 Simulation

We carry out simulations to study the performance of RDCM. First, we compare the performance of RDCM with three other representative reliable multicast schemes, namely, SRM [18], RMTP [19] and LBRM [21] (refer to Section 6 for the details of these protocols), in terms of application throughput and traffic footprint. Second, we study the effectiveness of the congestion control mechanism in RDCM as well as the impact of time to detect packet loss. We run all the simulations in a BCube(8,3) network, which is the size of a typical containerized data center supporting 4,096 servers. The speeds of all the physical links are 1Gbps.

4.1 Application Throughput and Traffic Footprint

In order to well demonstrate the performance differences among these reliable multicast protocols, we set a single multicast group in the network, and there is no background traffic. Since the congestion control mechanisms in the other protocols are not very clear, we do not use any congestion control to adjust the traffic sending rate in all the protocols including RDCM. In other words, the sender sends packets as fast as possible. Given multiple flows sharing a certain link (it occurs when the repair paths and the multicast tree are overlapped), the bottleneck link transmits in full speed, and all the flows equally share the bottleneck link bandwidth. The receivers dynamically join and leave the network, with a maximum group size of 1000. The sender and receivers of each group are randomly distributed in the network. The packet size in the multicast session is 1KB, and totally 1024 packets are sent out. Hence, the application traffic delivered is 1 MB.

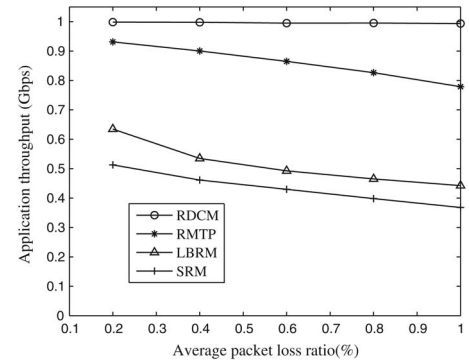


Fig. 10. Application throughput of the multicast session for different reliable multicast protocols in BCube network. RDCM outperforms other protocols by achieving almost full link capacity, due to the packet repair on backup overlay.

We measure two important performance metrics for these reliable multicast protocols, i.e., application throughput and traffic footprint. Application throughput is evaluated as the average goodput of the receivers. Traffic footprint is evaluated as the aggregated number of bytes on all the physical links, including the bytes of both successfully delivered packets and the traversing footprint of lost packets. In LBRM and RMTP, whether using multicast repair or unicast repair is determined by the number of receivers experiencing packet loss. We choose the repair technique (multicast or unicast) minimizing the repair traffic, instead of setting a fixed threshold. It is hence in favor of LBRM and RMTP in the simulation results. In LBRM, we use typical two-tier logging servers. Of course a reliable multicast protocol is better if it achieves higher application throughput and causes less traffic footprint given the same network environment.

The packet loss model we use is Gilbert model [33], since bursty packet loss is common in both Internet and data center networks [32], [12]. The burst length and burst density is set that the average packet loss ratio per link varies as 0.2%, 0.4%, 0.6%, 0.8% and 1%.

Application Throughput: The application throughputs of the multicast session for these reliable multicast schemes are shown in Fig. 10. the reliable multicast schemes, the application throughput decreases with higher packet loss ratio. At any packet loss ratio, RDCM performs best among all these schemes. For the cases we test, This result follows our design rationale. RDCM improves the application throughput by achieving repair isolation and duplicate repair avoidance, leveraging the rich link resource in data center networks. In all the cases we simulate, RDCM can utilize almost full link capacity for effective packet delivery, due to packet repair on backup overlay. RDCM can more than double the application throughput in SRM.

SRM gets the lowest application throughput, since any lost packet is retransmitted to the entire group, and thus many receivers will receive unnecessary repair packets. RMTP and

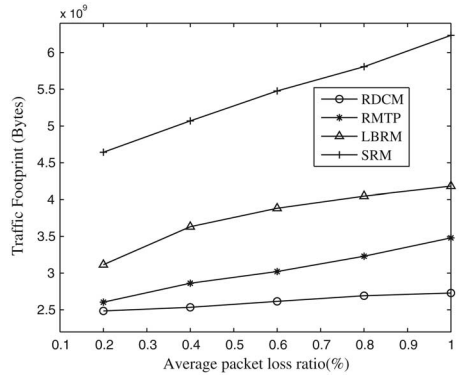


Fig. 11. Traffic footprint of the multicast session for different reliable multicast protocols in BCube network. RDCM has the lowest traffic footprint among all the protocols.

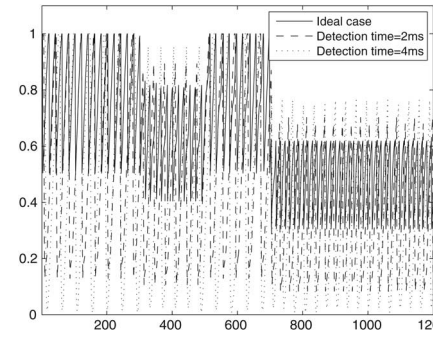


Fig. 12. The traffic sending rate in RDCM with congestion control. We test the ideal case when there is no delay to detect packet loss, and the cases when the timers are set as 2 ms and 4 ms respectively. Initially, there is no background traffic. At $t = 300s$, a UDP flow starts with 200 Mbps in a link in the multicast tree, and stops at $t = 500s$. At $t = 700s$, a UDP flow with 400 Mbps starts in another link of the multicast tree, and goes until the end of the session.

LBRM perform better than SRM, since they divide the whole multicast group into many hierarchical clusters for reliable data delivery; but worse than RDCM, because repair isolation and duplicate repair avoidance are not completely realized.

Traffic Footprint: Fig. 11 demonstrates the traffic footprints of these four reliable multicast schemes. RDCM uses unicast for packet repair. On the negative side, it increases the repair traffic for receivers simultaneously losing the same packet. But on the positive side, it achieves repair isolation and avoids multicasting the repair packets to receivers which do not observe packet loss. The simulation results show that the positive side outweighs the negative side. RDCM has the lowest traffic footprint among all these reliable multicast schemes.

SRM shows the highest traffic footprint. Its gap with other schemes increases when the packet loss ratio becomes higher. Though it uses multicast for retransmission, there are two reasons resulting in the high traffic overhead. First, even when only a few receivers lose a packet, the repair packet is multicast to the whole tree. Second, there is no repair isolation, hence the retransmission probability (given a certain packet loss ratio) in SRM is the largest among these schemes. The traffic footprints in LBRM and RMTP are between SRM and RDCM, since they use hierarchical clusters for reliable packet transmission.

4.2 Congestion Control in RDCM

We further evaluate the congestion control mechanism of RDCM, and consider the impact of the time to detect packet loss as well. Note that we use BIC for window adjustment. We run the whole multicast session for 1000 seconds. Initially, we set no background traffic in the network. At $t = 300s$, we generate a UDP flow with 200Mbps in a link in the multicast tree, and stop it at $t = 500s$. At $t = 700s$, we generate a UDP flow with 400Mbps in another link of the multicast tree, and let it go until the end of the multicast session.

The setting of the multicast session is as above, but we launch the congestion control in RDCM, which adjusts the sending rate at the multicast source according to network status. Note that the actual sending rate at the sender is determined by not only the congestion window, but also the queuing buffers in the intermediate switches. To simplify the situation, in our simulation we assume zero buffer in all the intermediate switches. Therefore, the average sending rate of the multicast session cannot make full usage of the available network bandwidth, since RDCM follows an AIMD control law.

As aforementioned, the timer length to detect packet loss will affect the responsiveness of senders to adjust the sending rate and accordingly the multicast throughput. In the simulation we test both the ideal cases when there is no delay to detect packet loss and the cases when the timers are set as 2 ms and 4 ms (the timer is relatively small since data transmission speed is very high in data centers).

Fig. 12 shows the sending rate at the sender. We find that the congestion control mechanism in RDCM can gracefully adapt to the network status. When packet loss occurs, the affected receivers reduce the congestion window and the sender adjusts the window to accommodate it. When the time required to detect packet loss is longer, there will be more packet losses and the congestion window drops to smaller value. Hence, the average application throughput of RDCM is lower when it takes more time to detect packet loss. But even with lags in loss detection, we do not observe congestion collapse in RDCM, due to the packet repair on the backup overlay, which usually has separate paths from the multicast tree.

5 Implementation and Experiments

In this section we present the implementation of RDCM as well as the experimental study on a test bed.

5.1 Implementation

We have implemented RDCM as a user-level library on Windows platform. Applications use UDP/IP sockets for multicast communication, which are intercepted by our library.

Fig. 13 shows the implementation architecture. The key components include repair window maintenance, interface to applications and message sending/receiving part interacting with Tree Manager as well as other data center servers. When the sender starts sending multicast packets to a group, the first packet is intercepted and a message is sent to the multicast manager. Data packets are sent out after the backup overlay is configured on all servers. Each packet is inserted with a

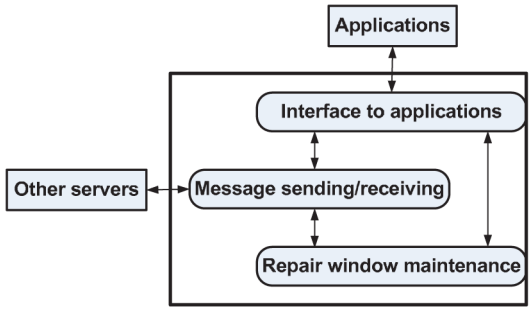


Fig. 13. Implementation architecture of RDCM.

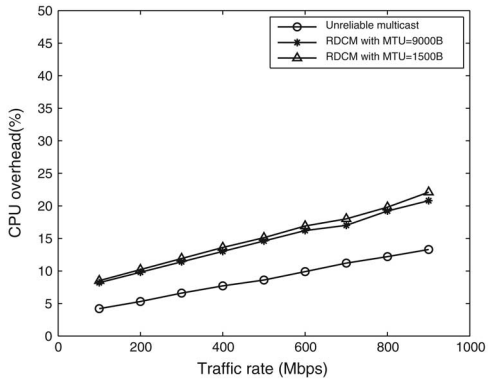


Fig. 14. CPU overhead of RDCM and unreliable multicast in multicast sessions with different data rates. The MTU is set by 1500 bytes and 9000 bytes (jumbo frame) respectively.

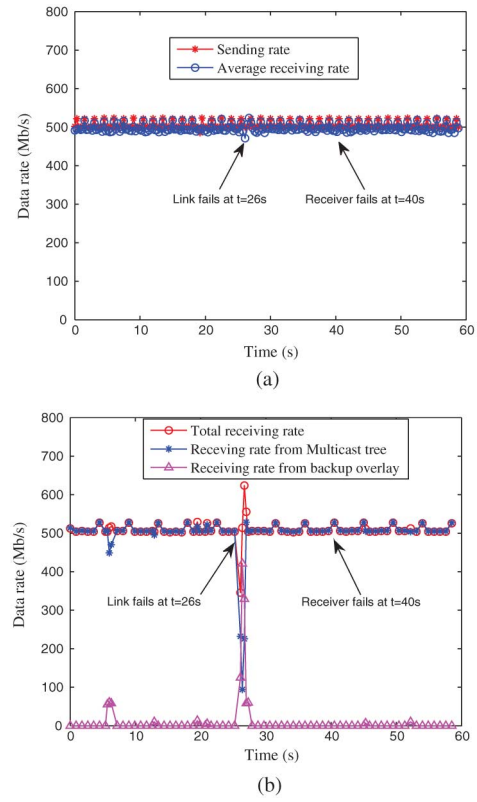


Fig. 15. Packet repair in RDCM. Link $w5 \rightarrow v5$ fails at $t = 26$ s. Receiver $v14$ fails at $t = 40$ s. (a) The data sending rate and average receiving rate of the multicast session; (b) The data receiving rate at $v5$.

sequence number. Receivers deliver received packets to upper-layer applications after removing the sequence number. The repair window is updated when receiving multicast packets from either the multicast tree or the backup overlay, or when receiving ACKs. In the implementation, we combine the ACKs for several packets into a single packet to reduce the processing overhead on servers. A receiver sends repair packets on the backup overlay when packet loss in its overlay successor is detected.

5.2 Experiments

We conduct experiments on a 16-server testbed. Each server has one Intel 2.33 GHz dualcore CPU, 2 GB DRAM, and an Intel Pro/1000 PT quad-port Ethernet NIC. The OS installed on all servers is Windows Server 2003 Enterprise x64 Edition. The 16 servers are interconnected as a BCube(4,1) topology, as shown in Fig. 4. Switches support Gigabit multicast forwarding.

In all our experiments, the sender is $v0$, and the receiver set is $\{v1, v2, v5, v8, v9, v10, v11, v12, v14\}$. The applications use legacy UDP/IP sockets for multicast communication. IP routing is used instead of BCube routing because current BCube routing does not support multicast. The MTU is set by 1500 bytes and 9000 bytes (jumbo frame) respectively. The timeout value to detect packet loss is set as 4 ms.

CPU Overhead: The sender generates multicast packets by CPU at different rates from 100Mbps to 900Mbps. For a certain data rate, we compare the average CPU usage among all receivers between RDCM and unreliable multicast, in which there is neither ACK exchange nor packet repair. Fig. 14 shows the results. In general, the CPU utilization with 1500 bytes is only marginally higher than with jumbo frame. Compared with RDCM, the additional CPU overhead brought by RDCM increases with the data rate, because of more frequent ACK processing and packet repair. But the additional CPU usage is always no higher than 10%. Even when the data rate is 900Mbps, the CPU usage in RDCM is only about 10% higher than that of unreliable multicast. When the data center group size is larger than our experiment setting, the CPU overhead is not expected to be much higher, because the overhead of ACK exchange and packet repair does not depend on the group size in RDCM. The only difference is that some receivers may receive ACKs from more overlay children in a larger group, but it will not pose much difference, because only the first ACK from overlay children for a packet is processed. The experiment results suggest that the overhead for maintaining backup overlay and repairing lost packets in the RDCM receivers is affordable on commodity servers.

Packet Repair: We set the sender to generate packets at about 500Mb/s and observe the effectiveness of packet repair in RDCM (without congestion control). The initial multicast tree is established as Fig. 7. At $t = 26$ s, we shutdown the link of $w5 \rightarrow v5$. At $t = 40$ s, we shutdown the receiver $v14$. Fig. 15(a) shows the data sending rate at the sender and the average receiving rate on all receivers. We find that the alive receivers can receive multicast data at the same speed of the sender throughout the whole session. The link failure and receiver failure have almost no impact on the multicast throughput. In our experiment, after we shutdown the link at t = 26s, link failure is quickly detected and identified, and the multicast tree is adjusted. During the switching interval,

Table 1
Comparison of Different Reliable Multicast Schemes

Scheme	SRM	LBRM	TMTP	RMTP	RDCM
Repair path	global tree	another tree	local tree	local tree	bypassing tree
Repair isolation	-	good when h^+ is large	fair	good when h is large	excellent
Request implosion avoidance	good	good	good	good	good
Duplicate repair avoidance	fair	excellent	excellent	excellent	excellent
Individual repair burden	lowest	low when h is small	lowest	low when h is small	low
Overall repair traffic	good under simultaneous loss	determined by h	good under simultaneous loss	determined by h	good
Repair latency	determined by back-off timer	low	determined by hierarchy levels	determined by hierarchy levels	fair

$^+$The threshold to decide whether using multicast repair or unicast repair.

the affected receiver ($v5$) receives packets from the backup overlay. When the receiver $v14$ fails at $\mathrm{t} = 40\mathrm{s}$, the other receivers are receiving the data from the multicast tree, and the backup overlay is quickly adjusted.

The data receiving rate at $v5$ is further inspected and demonstrated in Fig. 15(b). We separate the receiving rates from the multicast tree and from the backup overlay, the sum of which is the total receiving rate. We have the following observations from this figure. First, in the whole multicast session, there is no obvious degradation of the total receiving rate on $v5$, even when congestion or link failure occurs. Second, during the interval between link failure at $t = 26$ s and establishment of the new tree, $v5$ is receiving all packets from the backup overlay. But after $v5$ joins the new tree, it recovers receiving data from the multicast tree. Third, there is a spike on the total receiving rate after $v5$ joins the new tree, because in the short period after $v5$ joins the new tree, it is simultaneously receiving new packets from the multicast tree and lost packets from the backup overlay. Fourth, the failure of receiver $v14$ does not affect the receiving rate of $v5$, though they are in the same overlay ring in the backup overlay. It is because when $v14$ fails, $v5$ can receive data from the multicast tree, and the backup overlay is then quickly adjusted. Finally, there are always sporadic bursty packet loss on $v5$, in either the old multicast tree or the new multicast tree. The bursty length is usually 1-2 seconds. $v5$ gracefully handles this kind of light congestion by receiving lost packets from the backup overlay, without triggering tree adjustment. As a whole, the experiment demonstrates that RDCM responds gracefully and quickly to both link and receiver failures, and has no obvious impact on the application throughput.

6 Related Work

During the past two decades, many reliable Multicast solutions are proposed for the Internet. These proposals can be divided into two categories, namely, network-equipment based and end-host assisted. The former category is represented by PGM [22] and ARM [23], while the latter category includes SRM [18], RMTP [19], TMTP [20] and LBRM [21], etc. Since low-end commodity switches in data centers are not supposed to bear much intelligence, end-host based solutions can better accommodate data center network. In Table 1, we compare RDCM with the typical end-host based reliable Multicast solutions in terms of the important metrics, namely, packet repair path, repair isolation, request implosion avoidance, duplicate repair avoidance, individual repair burden, overall repair traffic and repair latency.

SRM is a reliable Multicast framework for light-weight sessions in the Internet, especially for distributed whiteboard application. LBRM is designed for distributed interactive simulation in the Internet, characterized by low data rate and requirement on realtime packet loss recovery. TMTP targets for Internet collaborative multimedia applications, while RMTP focuses on bulk data transfer in the Internet. RDCM differs from them in that it is especially designed to support data intensive group computations in data centers. Packet repair in RDCM is highlighted by *complete repair isolation and the capability to bypass the congested or failed tree link where packet loss occurs*. Hence, the impact of packet loss on the Multicast throughput is minimized.

Packet Repair Path: Repair packets in SRM, TMTP and RMTP all traverse along the Multicast tree. The difference is that in SRM, repair packet is Multicast to the whole group, while TMTP and RMTP use hierarchies and only local tree links are used for repair. LBRM provides a logging service, in fact another Multicast tree for packet repair, in order not to affect the data transmission in the main Multicast tree. RDCM repairs lost packets by Unicast, and has high probability to bypass the congested or failed tree links leveraging the rich connectivity in DCN and multiple paths between servers.

Repair Isolation: Repair isolation refers to the property that a repair packet is only sent to a local region where packet loss occurs. The ideal case is that a receiver receives a repair packet only if it has missed the packet. In SRM, the repair packet is Multicast to the whole group and how to realize repair isolation is an open issue. TMTP also uses Multicast for packet repair but limits the Multicast scope to the local domain. In LBRM and RMTP, either Unicast repair or Multicast repair is used based on the number of repair requests (or ACKs) received. If the number of receivers requiring packet repair exceeds a predefined threshold h, Multicast repair is used. Otherwise, multiple Unicast repairs are sent to corresponding receivers. Hence, repair isolation is good only when h is large. RDCM achieves complete repair isolation because Unicast repair is always chosen.

Request Implosion Avoidance: Request implosion means that a receiver/sender is overwhelmed by a large number of

ACKs/NAKs or repair requests for a single packet. All these reliable Multicast schemes perform well to avoid this problem. SRM and LBRM depend on a receiver to detect packet loss by itself and send explicit repair requests. SRM avoids request implosion by setting back-off timers on receivers. A triggered receiver Multicasts the request to the whole group, while other receivers hearing the request suppress their requests. LBRM builds a distributed logging service and distributes the repair requests for a single packet to many local logging servers. TMTP, RMTP and RDCM all construct a multi-level hierarchy structure, and each receiver sends positive ACK for a received packet to some designated receivers or sender. TMTP also lets a receiver Multicast an NAK to the local region so as to further suppress NAKs. As a result, in TMTP, RMTP and RDCM, a receiver/sender receives only a small number of ACKs/NAKs from other receivers.

Duplicate Repair Avoidance: Duplicate repair occurs when a receiver receives multiple repairs for a single lost packet. SRM tries to avoid this problem but cannot be guaranteed because of the probabilistic nature of the back-off algorithms. LBRM, TMTP, RMTP and RDCM all perform well in this respect since a designated receiver/sender is responsible for packet repair to a certain receiver.

Individual Repair Burden: Individual repair burden is measured by the maximum number of repair packets a receiver/sender sends for a single lost packet. SRM and TMTP are the lowest because they use Multicast for packet repair, and only one repair packet is sent. The individual repair burdens for LBRM and RMTP are low only when the threshold h is small, in which case a small number of Unicast repairs for a lost packet will be sent out by the designated receiver/sender. As a result, there is a tradeoff in LBRM and RMTP to set the threshold h. When h is small, the repair isolation is not good; but when h is large, the maximum individual repair burden is high. In RDCM, a receiver is responsible for repairing packets for at most *two* other receivers, no matter how large the group size is. The sender Multicasts a lost packet only when all receivers lose it. Consequently, individual repair burden in RDCM is very low.

Overall Repair Traffic: It is difficult to compare the overall repair traffic among these schemes. SRM and TMTP use Multicast for packet repair, which saves repair traffic when a large number of receivers simultaneously lose a packet, but transmits much unnecessary traffic when only a small set of receivers experience packet loss. The performance for LBRM and RMTP is determined by setting the threshold h of using whether Multicast repair or Unicast repair. RDCM saves repair traffic when the number of receivers experiencing simultaneous packet loss is small, because it performs best in repair isolation. But it can be less efficient when a large number of receivers lose the same packet. However, the repair traffic in RDCM is at most twice that of using Multicast repair. In fact we make a tradeoff in RDCM to enhance the Multicast throughput by leveraging the rich link resource of data center networks.

Repair Latency: The maximum repair latency in these schemes is often longer than that of source-to-end Unicast, because either back-off timer or hierarchy is used. The latency in SRM is determined by the back-off timer, which is usually multiple of the source-to-end Unicast delay. LBRM has low repair latency since only two levels of hierarchy is used. The maximum latency in TMTP and RMTP is determined by the number of hierarchy levels. Packet repair in RDCM usually traverses more hops than the other schemes, but the maximum repair latency is fair, as discussed in Section 3.

7 Conclusion

We presented RDCM, a dedicated reliable multicast proposal for data center networks. RDCM leverages the high link density in data centers to minimize the impact of packet loss on the throughput degradation of the multicast session. A multicast tree aware backup overlay is explicitly constructed upon group members for packet repair. When packet loss occurs, repair packet is transmitted in a peer-to-peer way on the backup overlay, which has high probability to bypass the congested/failed link in the multicast tree where packet gets lost. RDCM achieves complete repair isolation, which is critical for throughput enhancement. The backup overlay is carefully designed to control individual repair burden, control overhead as well as overall repair traffic. Congestion control and tree adjustment are also realized in RDCM. Simulations show that RDCM not only achieves higher application throughput than other representative reliable multicast schemes, but also brings less traffic footprint. The experiments on a 16-server testbed demonstrate that RDCM can reliably deliver high-speed multicast session by gracefully handling packet loss, link failure as well as receiver failure, while introducing low additional overhead.

Acknowledgment

The work was supported by the National Basic Research Program of China (973 program) under Grant 2014CB347800 and Grant 2009CB320501, the National Natural Science Foundation of China under Grant 61170291, Grant 61133006, and Grant 61161140454, and the National High-tech R&D Program of China (863 program) under Grant 2013AA013303. Some preliminary results of this work were published at IEEE INFOCOM'11, mini-conference. In this paper, we make substantial improvements based on the previous version, e.g., the multicast tree and backup overlay adjustment schemes, congestion control mechanisms, simulations to compare the performance of RDCM with other representative reliable multicast protocols, as well as more experiments.

References

[1] R. Bryant, "Data-intensive supercomputing: The case for DISC," Dept. Comput. Sci., CMU, Tech. Rep. CMU-CS-07–128, May 2007.
[2] M. Armbrust, A. Fox, R. Griffith et al., "Above the clouds: A Berkeley view of cloud computing," EECS Dept., UC Berkeley, Tech. Rep. UCB/EECS-2009–28, Feb. 2009.
[3] L. Barroso, J. Dean, and U. Holzle, "Web search for a planet: The Google cluster architecture," *IEEE Micro*, vol. 23, no. 2, pp. 22–28, Mar./Apr. 2003.
[4] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. Oper. Syst. Des. Implementation (OSDI'04)*, Dec. 2004, pp. 137–150.
[5] M. Isard, M. Budiu, Y. Yu et al., "Dryad: Distributed data-parallel programs from sequential building blocks," in *Proc. Eur. Conf. Comput. Syst. (EuroSys'07)*, Mar. 2007, pp. 59–72.
[6] Hadoop [Online]. Available: http://hadoop.apache.org, accessed on 2012.

[7] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," in *Proc. Symp. Oper. Syst. Principles (SOSP'03)*, Oct. 2003, pp. 29–43.
[8] A. Greenberg, J. Hamilton, N. Jain et al., "VL2: A scalable and flexible data center network," in *Proc. ACM SIGCOMM*, Aug. 2009, pp. 51–62.
[9] 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.
[10] D. Li, J. Yu, J. Yu et al., "Exploring efficient and scalable multicast routing in future data center networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM'11)*, Apr. 2011, pp. 1368–1376.
[11] 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.
[12] S. Kandula, S. Sengupta, A. Greenberg et al., "The nature of data-center traffic: Measurements & analysis," in *Proc. ACM SIGCOMM Internet Meas. Conf. (IMC'09)*, Nov. 2009, pp. 202–208.
[13] 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.
[14] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM*, Aug. 2008, pp. 63–74.
[15] 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.
[16] OpenFlow [Online]. Available: http://www.openflowswitch.org/, accessed on 2012.
[17] Y. Vigfusson, H. Abu-Libdeh, M. Balakrishnan et al., "Dr. multicast: Rx for data center communication scalability," in *Proc. ACM Eur. Conf. Comput. Syst. (EuroSys'10)*, Apr. 2010, pp. 349–362.
[18] 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.
[19] 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
[20] J. Griffioen and M. Sudan, "A reliable dissemination protocol for interactive collaborative applications," in *Proc. ACM Multimedia*, Nov. 1995, pp. 333–344.
[21] H. Holbrook, S. Singhal, and D. Cheriton, "Log-based receiver-reliable multicast for distributed interactive simulation," in *Proc. ACM SIGCOMM*, Oct. 1995, pp. 328–341.
[22] T. Speakman, J. Crowcroft, J. Gemmell et al., "PGM reliable transport protocol specification," RFC3208, Dec. 2001.
[23] L. Lehman, S. Garland, and D. Tennenhouse, "Active reliable multicast," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM'98)*, Mar. 1998.
[24] C. Guo, G. Lu, Y. Xiong et al., "Datacast: A scalable and efficient group data delivery service for data centers," Microsoft Tech. Rep. MSR-TR-2011–76, Jun. 2011.
[25] K. Obraczka, "Multicast transport mechanisms: A survey and taxonomy," *IEEE Commun. Mag.*, vol. 36, no. 1, pp. 94–102, Jan. 1998.
[26] B. Adamson, C. Bormann, M. Handley et al., "NACK-oriented reliable multicast (NORM) transport protocol," RFC3208, 2009.
[27] J. Chang and N. Maxemchuk, "Reliable broadcast protocols," *ACM Trans. Comput. Syst.*, vol. 2, no. 3, pp. 251–273, 1984.
[28] K. Birman, M. Handley, O. Ozkasap et al., "Bimodal multicast," *ACM Trans. Comput. Syst.*, vol. 17, no. 2, pp. 41–88, 1999.
[29] P. Eugster, R. Guerraoui, S. Handurukande et al., "Lightweight probabilistic broadcast," *ACM Trans. Comput. Syst.*, vol. 21, no. 4, pp. 341–374, 2003.
[30] M. Balakrishnan, K. Birman, A. Phanishayee et al., "Ricochet: Lateral error correction for time-critical multicast," in *Proc. USENIX Symp. Netw. Syst. Des. Implementation (NSDI'07)*, 2007, pp. 73–86.
[31] Y. Yang and S. Lam, "Internet multicast congestion control: A survey," in *Proc. Int. Conf. Telecommun. (ICT'00)*, 2000.
[32] M. Borella, D. Swider, S. Uludag et al., "Internet packet loss: Measurement and implications for end-to-end QoS," in *Proc. Int. Conf. Parallel Process. (ICPP'98)*, 1998.
[33] Telecom R&D, "Study of the relationship between instantaneous and overall subjective speech quality for time-varying quality speech sequences: Influence of the recency effect," ITU Study Group 12, contribution D.139, 2000.
[34] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proc. SIGCOMM*, 2011, pp. 350–361.
[35] *IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks*, IEEE Standard 802.1Q, 2005.
[36] K. Tan, J. Song, Q. Zhang et al., "A compound TCP approach for high-speed and long distance networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM'06)*, 2006, pp. 1–12.
[37] L. Xu, "Binary increase congestion control (BIC) for fast long-distance networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM'04)*, 2004, vol. 4, pp. 2514–2524.

Dan Li received the PhD degree in computer science from Tsinghua University, Beijing, China, in 2007. He is an associate professor with Computer Science Department, Tsinghua University. His research interest spans from Internet architecture and protocols, data center networking, and green networking.

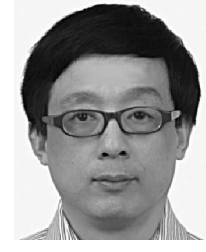
Mingwei Xu received the BSc and the PhD degrees in 1994 and 1998, respectively, from Tsinghua University, Beijing, China. He is a professor with the Department of Computer Science, Tsinghua University. His research interest includes future Internet architecture, Internet routing, and virtual networks.

Ying Liu received the MS degree in computer science and the PhD degree in applied mathematics from Xidian University, Xidian, China, in 1998 and 2001, respectively. She is currently an associate professor with the Tsinghua University, Beijing, China. Her research interests include network architecture design, multicast routing algorithm, and protocol.

Xia Xie photograph and biography not available at the time of publication.

Yong Cui received the PhD degree from Tsinghua University, Beijing, China, in 2004. He is now an associate professor in Tsinghua University. He is the winner of Best Paper Award of ACM ICUIMC 2011 and WASA 2010. His major research interests include mobile wireless Internet and computer network architecture.

Jingyi Wang is currently a master's student in Tsinghua University, Beijing, China.



Guihai Chen received the BS degree from Nanjing University, Jiangsu, China, in 1984, the ME degree from Southeast University, Nanjing, China, in 1987, and the PhD degree from the University of Hong Kong in 1997. He has a wide range of research interests with focus on sensor networks, peer-to-peer computing, massive data processing, high-performance computer architecture, and combinatorics. He has published more than 220 peer-reviewed papers, and more than 140 of them are in well-archived international journals such as *IEEE Transactions on Parallel and Distributed Systems*, *Journal of Parallel and Distributed Computing*, and also in well-known conference proceedings such as HPCA, MOBIHOC, INFOCOM, ICNP, ICPP, IPDPS, and ICDCS.