Hindawi Publishing Corporation
Advances in Multimedia
Volume 2007, Article ID 78913, 9 pages
doi:10.1155/2007/78913

Research Article

Scalable Island Multicast for Peer-to-Peer Streaming

Xing Jin,[1] Kan-Leung Cheng,[2] and S.-H. Gary Chan[1]

[1] *Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong*
[2] *Department of Computer Science, University of Maryland, College Park, MD 20742, USA*

Received 9 October 2006; Accepted 19 December 2006

Recommended by Guobin (Jacky) Shen

Despite the fact that global multicast is still not possible in today's Internet, many local networks are already multicast-capable (the so-called multicast "islands"). However, most application-layer multicast (ALM) protocols for streaming have not taken advantage of the underlying IP multicast capability. As IP multicast is more efficient, it would be beneficial if ALM can take advantage of such capability in building overlay trees. In this paper, we propose a fully distributed protocol called *scalable island multicast (SIM)*, which effectively integrates IP multicast and ALM. Hosts in SIM first form an overlay tree using a scalable protocol. They then detect IP multicast islands and employ IP multicast whenever possible. We study the key issues in the design, including overlay tree construction, island management, and system resilience. Through simulations on Internet-like topologies, we show that SIM achieves lower end-to-end delay, lower link stress, and lower resource usage than traditional ALM protocols.

1. INTRODUCTION

With the popularity of broadband Internet access, there has been increasing interest in media streaming services. Recently, peer-to-peer (P2P) streaming has been proposed and developed to overcome limitations in traditional server-based streaming. In a P2P streaming system, cooperative peers self-organize themselves into an overlay network via unicast connections. They cache and relay data for each other, thereby eliminating the need for powerful servers from the system. Currently, there are two types of overlays for P2P streaming: tree structure and gossip mesh. The first one builds one or multiple overlay tree(s) to distribute data among hosts. Examples include ALM protocols (e.g., Narada and NICE) and some P2P video-on-demand systems (e.g., P2Cast and P2VoD) [1–6]. The second one builds a mesh among hosts using gossip algorithms, with hosts exchanging data with their neighbors in the mesh [7–9]. The gossip-based approaches achieve high resilience to network and group dynamics. However, they have high control overhead due to data scheduling and mesh maintenance. They also have high playback delay because in the gossip mesh a host may not always find close peers as their neighbors. On the contrary, trees introduce lower end-to-end delay and are easier to maintain. Therefore, we consider a tree-based approach in this paper.

Most previously proposed ALM protocols (such as Narada, NICE, DT, Overcast, ALMI, etc.) assume that none of the routers are multicast-capable and do not take use of the underlying IP multicast capability [1, 2, 10–14]. Although global IP multicast is not available today, many local networks in today's Internet are already multicast-capable. These local multicast-capable domains, or so-called "islands," are often interconnected by multicast-incapable or multicast-disabled routers. Since IP multicast is more efficient than ALM, it would be beneficial if ALM makes use of the local multicast capabilities in building trees.

This integration is especially important for streaming applications. Let *link stress* be the number of copies of a packet transmitted over a certain physical link [1]. We have done simulations on Internet-like topologies to evaluate some representative ALM protocols. In a group of 1024 hosts, the average link stresses achieved by Narada [1], GNP-based DT [15], TAG [13], and Overcast [11] are 2.9, 2.69, 2.61, and 2.02, respectively. The maximum link stresses achieved by these protocols are 40, 24, 28, and 14, respectively. In other words, if we use Narada for streaming, each underlay link averagely needs to deliver 2.9 streams and the most loaded link has to deliver 40 streams. Considering that a single stream usually requires several hundred Kbps transmission rate, the current Internet often cannot provide enough bandwidth for the streaming. On the other hand, as IP multicast always keeps

the stresses of all the delivery links being 1, it significantly improves the delivery efficiency and reduces the bandwidth consumption. We hence propose a distributed and scalable protocol called *scalable island multicast (SIM)* that combines IP multicast with ALM for media streaming.

In SIM, hosts within an island exchange data with IP multicast. They connect across islands with unicast overlay paths. Each host first distributedly joins an overlay tree. It then detects and joins its multicast island if possible. Each island in SIM has a unique ingress host. The ingress receives streaming data from outside of the island and IP multicasts them within the island. Other hosts within the island receive data from IP multicast instead of from their parents in the overlay tree. We study the following key components in SIM.

(i) Construction of the overlay tree

We design a fully distributed host-joining mechanism for the construction of the overlay tree. A new host can iteratively ping other hosts and select a close peer with enough forwarding bandwidth as the parent.

(ii) Island management

Hosts within the same multicast domain form an island. Each island has two multicast groups: a CONTROL group and a DATA group. The control messages (e.g., for ingress selection) are only multicasted in the CONTROL group, and the streaming data are multicasted in the DATA group. We further study the ingress selection and island detection issues in detail.

(iii) System resilience

A single delivery tree may not provide good streaming quality, especially in a dynamic P2P system. We discuss two possible ways to improve system resilience: (1) build multiple trees for data delivery, and (2) use a quick recovery scheme to address temporary packet loss. In the recovery scheme, each host selects a few recovery neighbors under the constraint of the recovery deadline. Whenever a packet loss is detected, the retransmission request is sent to the recovery neighbors.

We have evaluated SIM through simulations on Internet-like topologies. Our simulation results show that SIM can efficiently combine IP multicast with ALM to achieve low end-to-end delay, low link stress, and low resource usage.

The rest of the paper is organized as follows. In Section 2, we briefly review the related work. In Section 3, we describe the data delivery mechanism in SIM. In Section 4, we discuss how to improve system resilience. In Section 5, we present illustrative simulation results. We finally conclude in Section 6.

2. RELATED WORK

We briefly review previous work on island multicast as follows. Though protocols such as mTunnel, scattercast, YOID, UMTP, AMT, and universal multicast (UM) have been proposed to combine IP multicast with ALM, many of them require special nodes (such as proxies or servers) or manual configuration for interhost connections [16–19]. SIM is fully autonomous and does not require any special or super nodes. Subset multicast (SM) also makes use of local multicast capability [20]. However, it is based on a star topology and is not scalable to large groups. The work in [21] proposes a centralized island multicast algorithm. As opposed to them, SIM is fully distributed and scalable. The work in [22] studies a distributed approach to integrate IP multicast and ALM. Each island has a leader, which identifies some ingress and egress hosts in its island for data delivery. This approach puts heavy control loads to leaders and has complicated mechanism for the management of leaders, ingress hosts, and egress hosts. SIM provides a much simpler data-delivery method and is much more implementable. In SIM, there is no leader, and there is no overhead to select egress hosts.

In HMTP, each island has a unique leader (called *designated member*, or *DM*) [23]. DMs form an overlay tree for data delivery. Each DM then IP multicasts data within its island. While HMTP imposes the responsibilities of data receiving, data forwarding, and island management on a single leader in each island, a leader has high nodal stress and heavy workload. Different from it, SIM distributes these responsibilities to different hosts. Each island in SIM has one ingress and some egress hosts. SIM hence achieves a more balanced load distribution. Furthermore, when islands are large, it is not efficient to represent each island by a single DM, where end-to-end delays depend on the locations of DMs and the selection of appropriate DMs is not easy. In SIM, we can select a close pair of hosts to connect two islands. This method is more efficient and practical. Another important difference between HMTP and SIM is the construction of the overlay tree. In HMTP, a new host starts joining the tree from the tree root in a top-down manner. The top-level hosts in the tree (i.e., those close to the root) are frequently visited by new hosts and are easily overloaded by the ping requests. It is hence not fully scalable. In SIM, each new host obtains a list of randomly selected peers at the beginning and starts joining from these hosts. The communication overhead for joining is hence distributed to all the peers.

A preliminary version of SIM has been discussed in [24]. In this paper, we further propose a loss-recovery scheme to improve streaming quality and conduct more comprehensive simulations to evaluate SIM.

3. DATA DELIVERY IN SIM

In this section, we describe the data delivery mechanism in SIM. Hosts first form a low-delay overlay tree. They then detect multicast islands and use IP multicast if possible. Note that unicast connections in SIM use TCP. But IP multicast within an island uses UDP, for TCP is not available in this case. In the following discussion, a *parent* of a host refers to the host's parent in the overlay tree, and the *source distance* of a host refers to the delay between the source and the host in the overlay tree.

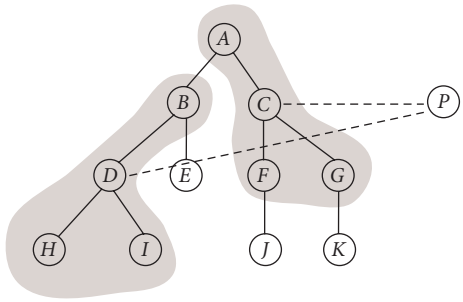


Figure 1: An example of joining the overlay tree in SIM.

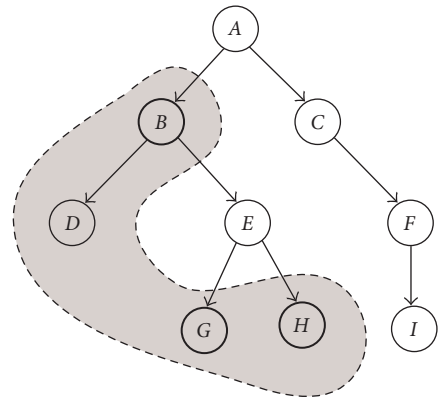


Figure 2: Ingress selection in a SIM tree.

3.1. Construction of the overlay tree

We are interested in building a tree with low end-to-end delay. Clearly, the tree construction mechanism should be distributed and scalable, and the algorithm should be simple with low setup and maintenance overhead.

In SIM, a new host first contacts a public rendezvous point (RP) to obtain a list of current hosts in the system. It pings these hosts and selects k closest ones. Then, it pings the neighbors of these selected hosts, and selects k closest ones from all the hosts (the original k hosts and their neighbors). This process is repeated until the improvement on round-trip time (RTT) is lower than a certain threshold, or the number of iterations exceeds a certain value t. At the end of the process, the new host selects from its current k closest hosts the one with enough forwarding bandwidth as its parent. If there are no qualified hosts, the new host goes back up one level to look for qualified parents.

Figure 1 shows an example of host joining in our scheme. Suppose $k = 2$ and P is a new host. P first obtains a list of hosts from the RP, say, C, D, E, F, and G. P then pings all of these hosts and selects two closest ones, say C and D. P then pings all of C's and D's neighbors. It continues selecting two closest hosts from C, D, C's neighbors (i.e., A, F, and G), and D's neighbors (i.e., B, H, and I). Such iteration stops if any of the above stopping rules is satisfied. Since the list of hosts returned by the RP is randomly generated, the communication overhead for joining is distributed to all the hosts.

If a host leaves, its children need to rejoin the tree and find new parents. A rejoining host starts rejoining from its grandparent and then acts as joining.

3.2. Integrating IP multicast

After a host joins the overlay tree, it detects its island and joins the island if any. Each streaming session has two unique class-D IP addresses for IP multicast. One is used for multicasting control messages, and the other is used for multicasting streaming data. We call the groups corresponding to these two IP addresses a *CONTROL group* and a *DATA group*, respectively.

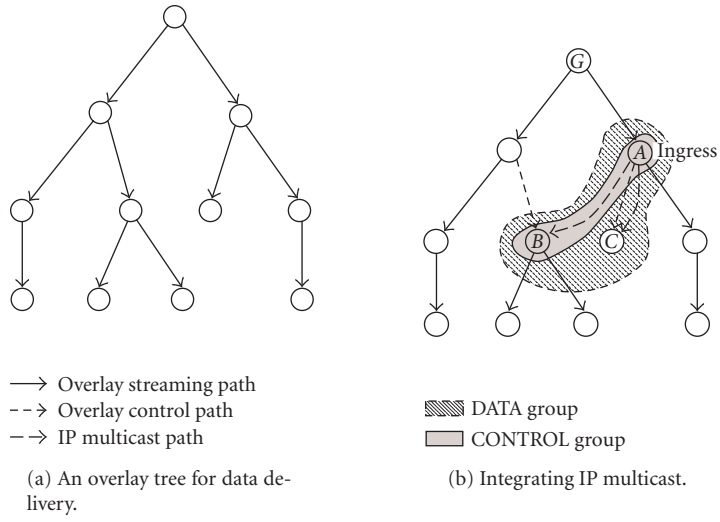
Each island has a unique ingress host, which is responsible for accepting streaming data from outside of the island and multicasting them within the DATA group. Other hosts within the island accept streaming data from IP multicast instead of overlay unicast. We call a host within the island a *border host* if its overlay parent is not within the island. Clearly, the ingress must be a border host. In SIM, border hosts (including the ingress) join both the CONTROL group and the DATA group, and nonborder hosts only join the DATA group.

3.2.1. Selection of the ingress

Proper selection of the ingress is important for efficient data distribution. If an ingress has a high source distance, all the hosts within its island will accordingly suffer high-source distances. Furthermore, not every border host can serve as an ingress. For example, in Figure 2, hosts first form an overlay tree. Later on, it is detected that B, D, G, and H are within the same island, and among them B, G, and H are border hosts. However, among the three border hosts, only B can serve as the ingress. If G becomes the ingress, all the other hosts within the island will stop receiving data from their overlay parents. When B is waiting for the data IP multicasted by G, E cannot receive any data from B. Consequently, G cannot receive any data from E. A deadlock is hence formed. Similarly, H cannot serve as the ingress.

To address these problems, we require each host to record its own source distance. The distance can be computed along the tree in a top-down manner. Namely, the source distance of a host is equal to the sum of its parent's source distance and the delay from its parent to itself. An ingress is hence selected from the border hosts of the island as the one with the minimum source distance.

An ingress host periodically multicasts *KeepAlive* messages in the CONTROL group, which contains its source distance. It also multicasts streaming data within the DATA group. Initially, the ingress of an island is the island's first joining host. The noningress border host with the smallest source distance in the island becomes the new ingress



(a) An overlay tree for data delivery.

(b) Integrating IP multicast.

Figure 3: Combining IP multicast with ALM.

if

(i) the current ingress leaves or fails (detected through the missing of *KeepAlive* messages), or
(ii) the border host has a smaller source distance than the current ingress.

This can be achieved as follows. A border host may multicast its own source distance within the CONTROL group. When a border host finds that its own source distance is smaller than the smallest source distance being multicasted, it keeps multicasting its own source distance within the CONTROL group. On the other hand, if a border host receives a message reporting a smaller source distance than its own one, it stops multicasting its own source distance. In the end, only the border host with the smallest source distance will multicast its source distance. If this source distance is smaller than that of the ingress, the corresponding border host will substitute the current ingress.

3.2.2. Island detection

The two class-D IP addresses are maintained by the RP. When a new host joins the session, it first obtains the class-D addresses and a list of already joined hosts from the RP. The new host then joins the overlay tree as described above. Afterwards, it joins the CONTROL group.

(i) If an island exists, the host will receive *KeepAlive* messages from the ingress. The host then detects whether itself is a border host. If it is, it remains in the CONTROL group and further joins the DATA group. Otherwise, it exits the CONTROL group and only joins the DATA group.

The examination of whether being a border host can be achieved as follows. The host multicasts a *BorderIdentification* message within the CONTROL group. If its parent receives the message, the parent unicasts a response message to the host using TCP. If the host does not receive any response after a certain time, it classifies itself as a border host.

A noningress host in the DATA group stops receiving streaming data from its overlay parent. Instead, it accepts data transmitted by IP multicast. The connection to its overlay parent is then only used for transmitting control messages. If this host becomes an ingress later, it resumes the overlay connection and accepts data from its parent again.

(ii) If the host does not find any island to join, it forms an island only consisting of itself and becomes the island ingress.

We show an example of data delivery with IP multicast in Figure 3. Figure 3(a) shows the overlay tree formed as described above. In Figure 3(b), hosts A, B, and C join the CONTROL group and detect that they are within the same island. A is elected as the island ingress. B is a normal border host and C is a nonborder host. Therefore, A and B stay in both the CONTROL group and the DATA group, while C only stays in the DATA group. A then accepts data from its overlay parent G, and multicasts them within the DATA group. The incoming overlay paths of B and C are then used for delivering control messages instead of media contents. If A leaves the system, B will be elected as the new ingress since it is the only border host within the island. B will then resume data delivery along its overlay path and multicast data within the island.

4. DISCUSSION ON SCHEME EXTENSION

In this section, we discuss two possible ways to improve system resilience, that is, building multiple trees and conducting packet-loss recovery.

4.1. Building multiple trees

Using a single tree may not offer satisfactory service, because, firstly, hosts in the system are heterogeneous with different

incoming and outgoing bandwidth. A host's incoming path may not be able to provide enough bandwidth for streaming. Secondly, quality degradation at a host (e.g., packet loss or host failure) affects all its descendants. In a highly dynamic P2P system, it is difficult for hosts to achieve high streaming quality with a single tree. To address these problems, we can use multiple description coding (MDC) [25] to encode streaming data into multiple descriptions and distribute the descriptions along multiple trees [5].

In MDC, data are encoded into several descriptions. When all the descriptions are received, the original data can be reconstructed without distortion. If only a subset of the descriptions are received, the quality of the reconstruction degrades gracefully. The more descriptions a host receives, the lower distortion the reconstructed data have. Therefore, the source can encode its media content into M descriptions using MDC (where M is a tunable parameter), and transmit the descriptions along M different trees. Note that a host has different descendants in different trees. The descendant of a host in one tree is usually not the host's descendant in other trees. Therefore, packet loss at a host or failure of the host only causes the loss of a single description (out of M descriptions) at each of its descendants. The system resilience is hence improved.

4.2. Packet-loss recovery

Although MDC and multiple-tree transmission can improve resilience, packets may still get lost due to background traffic or path/host failure. An efficient loss recovery mechanism is hence desired to deal with temporary packet loss.

4.2.1. Design principle

Traditional source-recovery and parent-recovery schemes have loss correlation problem and implosion problem [26]. That is, the losses of all downstream hosts are correlated upon an upstream loss. The parent of a failed host is hence likely in error, leading to low retransmission efficiency. Furthermore, such recovery leads to implosion if retransmission requests from downstream hosts are not aggregated (which is often the case for simplicity). To address these problems, lateral error recovery (LER) has been proposed [26, 27]. LER randomly divides hosts into multiple planes and independently builds an overlay tree in each plane. A host needs to identify some hosts from other planes as its recovery neighbors. Whenever a loss occurs, the host performs retransmission from its recovery neighbors. A limitation of LER is that the trees in different planes should be of similar sizes. Otherwise, a host in a small tree needs to serve as the recovery neighbor of multiple hosts in a large tree and has high workload. However, the balancing of multiple trees in a dynamic system requires high control overhead and is not easy.

We consider simplifying the selection of recovery neighbors as follows. We only use a single plane instead of multiple planes. The recovery neighbor of a host should satisfy the following requirements: (1) not in the host's subtree; (2) not the ancestor of the host; and (3) not in the same island as the host. Clearly, the loss correlation between a host and its recovery neighbor in this scheme is higher than that in LER. In LER, the path from the root to a host and the path from the root to the host's recovery neighbor are disjoint on the overlay. But this is not true in our scheme. However, our scheme does not need to balance multiple trees, thereby introducing much lower control overhead.

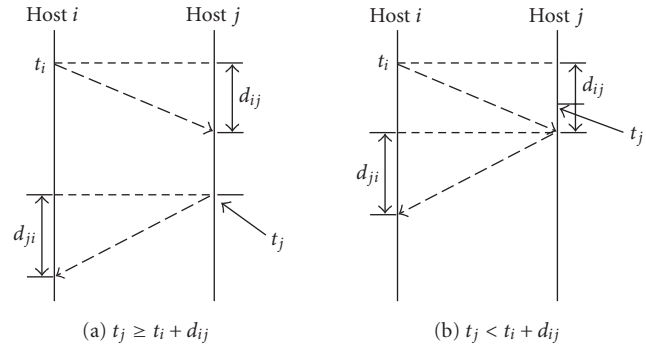


Figure 4: Time diagram for recovery neighbor selection.

Recovery neighbors are useful for improving streaming quality. For example, if the path between a host and its parent does not have enough bandwidth, the host can require missing data from one or multiple of its recovery neighbors. This enables multiple-path delivery as in gossip streaming.

4.2.2. Selection of recovery neighbors

We have listed three basic requirements for selecting recovery neighbors as above. However, to achieve quick loss recovery, more careful selection should be considered. Firstly, a streaming application usually has a certain recovery deadline δ after detecting a loss. Define *recovery latency* as the time interval from the moment a loss is detected to when the repair is received. The recovery latency should be smaller than δ. A host hence cannot select a faraway host as its recovery neighbor, for the retransmission time may exceed the recovery deadline. Secondly, the lost packet at a host is not available at an arbitrary host in the system. If the buffers of hosts have finite sizes, it may happen that the packet needed is no longer in the buffers. Even if the buffers are infinitely large and hosts cache all the data they have received, it may happen that hosts have different end-to-end delays and the requested data have not arrived at all the hosts in the system. In LER, it is assumed that the buffers of hosts are all infinitely large, which is often not true in real systems. We hence study how to select an appropriate recovery neighbor with limited buffer sizes.

Let t_i and t_j be the source distances of host i and j, respectively, where j is a recovery neighbor of i. Let d_{ij} and d_{ji} be the one-way delay from i to j and from j to i, respectively. Let B_i and B_j be the buffer sizes of i and j (in terms of time), respectively. As usual, we assume $B_x \geq \delta$ for any host x in the system. Figure 4 shows the time diagram upon a loss detected at time t_i at host i, given that the packet is transmitted from the source at time 0. When i requests a retransmission

Table 1: Requirements on host j as a recovery neighbor of host i.

Type	Condition	Recovery latency	Requirements
I	$t_j \geq t_i + d_{ij}$	$t_j + d_{ji} - t_i$	$t_j + d_{ji} - t_i \leq \delta$
II	$t_j < t_i + d_{ij}$	$d_{ij} + d_{ji}$	$d_{ij} + d_{ji} \leq \delta$, $t_i + d_{ij} \leq t_j + B_j$

from host j at time t_i, the retransmission request arrives at j at time $t_i + d_{ij}$.

Figure 4(a) shows the case where the arrival of the retransmission request is no later than the arrival of the requested packet, that is, $t_j \geq t_i + d_{ij}$. The recovery latency is hence $t_j + d_{ji} - t_i$. Clearly, the buffer size does not matter in this case, and the only requirement is $t_j + d_{ji} - t_i \leq \delta$. Figure 4(b) shows the case where the arrival of the retransmission request is later than the arrival of the requested packet, that is, $t_j < t_i + d_{ij}$. The recovery latency is hence $d_{ij} + d_{ji}$. Clearly, it is required that $d_{ij} + d_{ji} \leq \delta$. Furthermore, upon the arrival of the retransmission request, the requested packet should still be in host j's buffer, that is, $t_i + d_{ij} \leq t_j + B_j$. Table 1 summarizes these requirements on qualified recovery neighbors. Among all the qualified candidates, it is better to select the ones with high-bandwidth connections. If no qualified recovery neighbor can be found, we select from Type-I hosts the ones with low recovery latency.

The detailed selection process works as follows. As discussed in Section 3.2.1, each host has recorded its source distance (i.e., t_i and t_j). For simplicity, we assume that d_{ij} and d_{ji} are equal to half of the RTT from i to j and from j to i, respectively. A host i first joins the overlay tree for data delivery. It then contacts its ancestors within L_1 hops. In each contact, i checks the ancestor's descendants within L_2 hops. Here L_1 and L_2 are two system parameters. If the host checked satisfies the above requirements (including the three basic principles), i adds the host as a candidate of its recovery neighbor. If the requirements in Table 1 cannot be satisfied but the host is of Type-I, i adds the host to a list of Type-I hosts. After collecting a certain number of recovery neighbors and Type-I hosts, i stops the selection process. Otherwise, i increases L_1 and L_2 and keeps selecting recovery neighbors.

5. ILLUSTRATIVE SIMULATION RESULTS

In this section, we present illustrative simulation results on Internet-like topologies.

5.1. Simulation setup

We generate 10 *transit-stub* topologies with GT-ITM [28]. Each generated topology is a two-layer hierarchy of transit domains and stub domains. The transit domains form a backbone and all the stub domains are connected to the backbone. In our simulations, each topology has 4 transit domains and 280 stub domains. On average, a transit domain contains 10 routers and a stub domain contains 8 routers. Each topology consists of 2280 routers and about 11 000 links. A group of 1024 hosts are randomly put into the network. A host is connected to a unique stub router with 1 millisecond delay, while the delays of core links are given by the topology generator. Link bandwidth is set as follows: a backbone link (at least one end-point is a transit router) can support 8 concurrent media streams, and a nonbackbone link can support 3–6 concurrent media streams. The distribution of islands is set as follows: from the stub domains that consist of at least one host, we randomly select some and set them to be multicast-capable. Define *multicast ratio* θ as the ratio of the number of multicast-capable stub domains to the number of stub domains that consist of at least one host, and define *island size* S as the number of stub domains in an island. Note that in the real Internet, routers in a multicast island are often close to each other. Therefore, in our simulations, only the stub domains connected to the same transit router can be within the same multicast island.

The SIM parameters are set as follows. Each new host obtains a number of (at most 10) randomly selected hosts from the RP when joining. A new host repeats the ping iterations for at most 6 times and in each iteration pings at most 10 hosts (i.e., $t = 6$ and $k = 10$). We further implement two tree-based ALM protocols for comparison, that is, Narada [1] and Overcast [11]. Narada is one of the pioneering ALM protocols and aims at building a low-delay overlay tree. Its performance can serve as the benchmark. In Narada, each host has a degree bound according to its edge bandwidth. Overcast aims at constructing a tree with high bandwidth, which achieves low stresses on links.

We use the following metrics to evaluate the protocols.

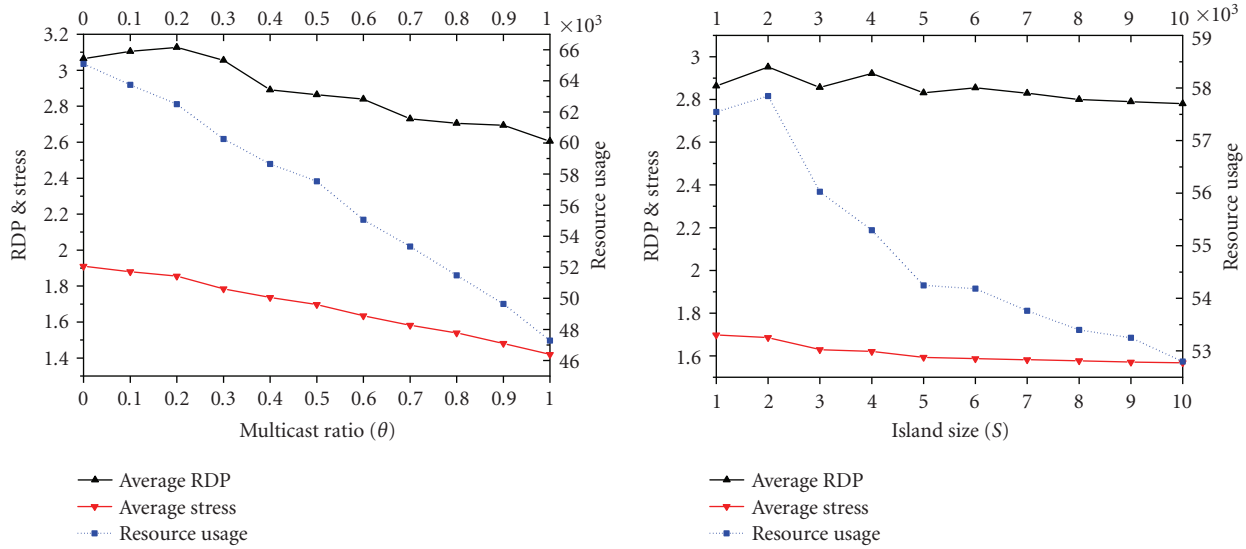
(i) *Relay delay penalty (RDP)*: defined as the ratio of the overlay delay from the source to a given host to the delay between them along the shortest unicast path [1].

(ii) *Link stress*: defined as the number of copies of a packet transmitted over a certain physical link [1].

(iii) *Resource usage*: defined as $\sum_{i=1}^{L} d_i \times s_i$, where L is the number of links active in data transmission, d_i is the delay of link i, and s_i is the stress of link i [1]. Resource usage is a metric of the network resource consumed in data delivery.

5.2. Results

Figure 5 shows the performance of a SIM tree with different parameter settings. Figure 5(a) shows the result of tuning the multicast ratio θ. As θ increases, the average RDP first increases and then decreases. This is because two hosts within the same multicast island are not necessarily close to each other, therefore selecting a host from other domains as the parent may introduce lower delay than simply receiving IP multicast packets from the ingress. However, when most hosts in the system are within multicast islands, the distance between a host and its ingress is often not large. The average RDP is hence low. On the other hand, both the average stress and the resource usage decrease as θ increases. When θ increases from 0 to 1.0, the average stress and the resource usage are reduced by 25.7% and 27.4%, respectively. Therefore,



(a) Performance of SIM with different multicast ratios ($S = 1$).

(b) Performance of SIM with different island sizes ($\theta = 0.5$).

Figure 5: Performance of SIM with different parameter settings.

the utilization of IP multicast in ALM can efficiently reduce the loads on links and the overall bandwidth consumption.

Figure 5(b) shows the results of tuning the island size S. As S increases, the average RDP fluctuates around 2.85. Averagely, a maximum island in our simulations can contain 10 stub domains (i.e., $S = 10$). In fact, from our simulation results, the average number of hosts in an island when $S = 10$ is around 30. Therefore, the islands formed are small as compared to the group size and such small islands cannot significantly reduce the end-to-end delay. We expect lower RDP when the islands are larger. Different from the average RDP, the average stress decreases as S increases. Clearly, IP multicast always achieves an average stress of 1.0. The more IP multicast paths in the delivery tree, the lower average stress. On the other hand, the resource usage first slightly increases and then decreases when S increases. It shows that the penalty in delay exceeds the improvement in stress when S increases from 1 to 2. When S continues increasing, the improvement in stress leads to the reduction in the resource usage.

Figure 6 compares the performance of different ALM protocols. We select a set of (θ, S) combinations for SIM. Figure 6(a) shows the average RDP achieved by different protocols. Overcast has the highest RDP among the protocols. This is because it only aims at improving tree bandwidth and does not optimize the tree in terms of end-to-end delay. Furthermore, to reserve bandwidth for future hosts, each host in Overcast is inserted into the tree as far from the source as the bandwidth constraint allows. It hence performs poorly in terms of RDP. SIM$(0,*)$ shows the result with no multicast islands. It performs slightly worse than Narada. When θ is 0.5, the average RDP of SIM is lower than that of Narada. Furthermore, when $\theta = 1$ and $S = 10$, all the hosts are within islands and the average RDP of SIM is significantly reduced.

Figure 6(b) compares the average stress achieved by different protocols. Narada has the highest average stress. Overcast targets maximizing bandwidth and accordingly minimizes link stress. It hence has lower average stress. SIM shows very low average stress, even in the absence of multicast islands. With the increase of θ and S, the average stress of SIM decreases. When $\theta = 1$ and $S = 10$, the average stress is only 1.14, which is very close to the optimal value of 1.0. The maximum stresses achieved by the protocols show similar trends as the average stresses (as shown in Figure 6(c)). The utilization of IP multicast can efficiently reduce the maximum stress of SIM, from 19.4 (when $\theta = 0$) to 7.5 (when $\theta = 1$ and $S = 10$). Clearly, lower stresses indicate lighter loads on links and hence higher transmission rates. From the figures, SIM can achieve low stresses and is efficient for streaming applications.

Figure 6(d) shows the resource usage achieved by different protocols. Since Narada has the highest stress and Overcast has the highest RDP, their resource usage is high. SIM achieves good tradeoff between RDP and link stress. Its resource usage is significantly lower than Narada and Overcast.

6. CONCLUSION

Traditional ALM protocols only make use of unicast connections to form delivery trees and have not fully taken advantage of the local multicast capabilities. In this paper, we propose a fully distributed multicast protocol (called SIM) for media streaming which combines IP multicast with ALM. Hosts in SIM can distributedly detect multicast domains and use IP multicast if possible. Simulations results show that it can achieve low end-to-end delay, low link stress, and low resource usage.

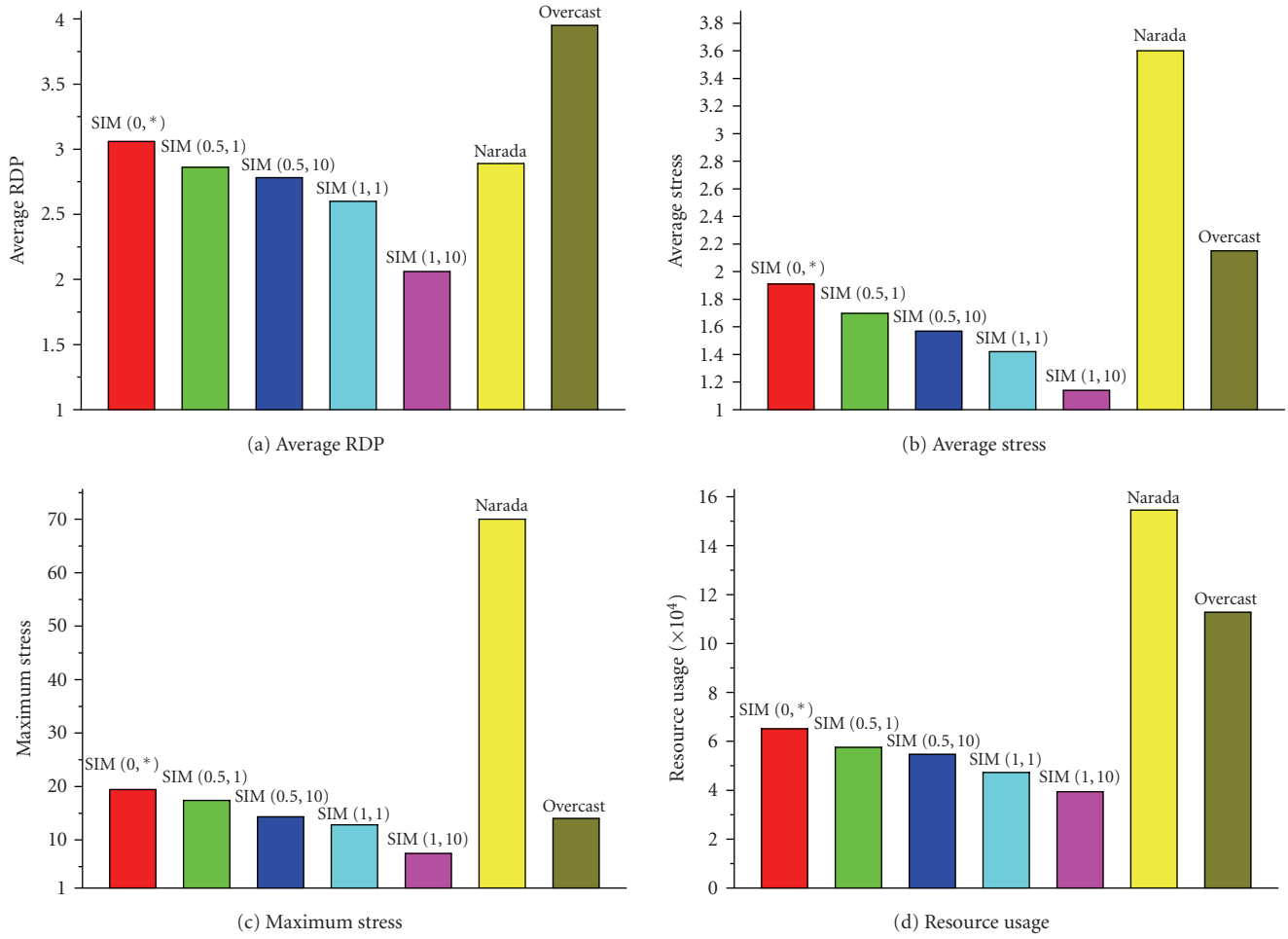


Figure 6: Performance comparison of different ALM protocols.

ACKNOWLEDGMENTS

This work was supported, in part, by Direct Allocation Grant (DAG05/06.EG10) and the Innovation and Technology Commission of the Hong Kong Special Administrative Region, China (GHP/045/05).

REFERENCES

[1] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1456–1471, 2002.

[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proceedings of the Annual Conference of the Special Interest Group on Data Communication (SIGCOMM '02)*, pp. 205–217, Pittsburgh, Pa, USA, August 2002.

[3] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: peer-to-peer patching scheme for VoD service," in *Proceedings of the 12th International World Wide Web Conference (WWW '03)*, pp. 301–309, Budapest, Hungary, May 2003.

[4] T. T. Do, K. A. Hua, and M. A. Tantaoui, "P2VoD: providing fault tolerant video-on-demand streaming in peer-to-peer environment," in *Proceedings of IEEE International Conference on Communications (ICC '04)*, vol. 3, pp. 1467–1472, Paris, France, June 2004.

[5] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: high-bandwidth multicast in cooperative environments," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, pp. 298–313, Lake George, NY, USA, October 2003.

[6] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, pp. 282–297, Lake George, NY, USA, October 2003.

[7] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, vol. 3, pp. 2102–2111, Miami, Fla, USA, March 2005.

[8] M. Zhou and J. Liu, "A hybrid overlay network for video-on-demand," in *Proceedings of IEEE International Conference on Communications (ICC '05)*, vol. 2, pp. 1309–1313, Seoul, Korea, May 2005.

[9] GridMedia. http://www.gridmedia.com.cn/.

[10] J. Liebeherr, M. Nahas, and W. Si, "Application-layer multicasting with Delaunay triangulation overlays," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1472–1488, 2002.

[11] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, "Overcast: reliable multicasting with an overlay network," in *Proceedings of the 4th Symposium on Operating System Design and Implementation (OSDI '00)*, pp. 197–212, San Diego, Calif, USA, October 2000.

[12] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: an application level multicast infrastructure," in *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS '01)*, pp. 49–60, San Francisco, Calif, USA, March 2001.

[13] M. Kwon and S. Fahmy, "Topology-aware overlay networks for group communication," in *Proceedings of the 12th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '02)*, pp. 127–136, Miami, Fla, USA, May 2002.

[14] X. Jin, Y. Wang, and S.-H. Gary Chan, "Fast overlay tree based on efficient end-to-end measurements," in *Proceedings of IEEE International Conference on Communications (ICC '05)*, vol. 2, pp. 1319–1323, Seoul, Korea, May 2005.

[15] W.-C. Wong and S.-H. Gary Chan, "Improving delaunay triangulation for application-level multicast," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '03)*, vol. 5, pp. 2835–2839, San Francisco, Calif, USA, December 2003.

[16] P. Parnes, K. Synnes, and D. Schefström, "Lightweight application level multicast tunnelling using mTunnel," *Computer Communications*, vol. 21, no. 15, pp. 1295–1301, 1998.

[17] Y. Chawathe, *Scattercast: an architecture for internet broadcast distribution as an infrastructure service*, Ph.D. thesis, University of California, Berkeley, Calif, USA, 2000.

[18] P. Francis, P. Radoslavov, R. Lindell, and R. Govindan, "Your Own Internet Distribution YOID," http://www.isi.edu/div7/yoid/, 2002.

[19] R. Finlayson, "The UDP multicast tunneling protocol," draft-finlaysonumtp-09.txt, November 2003.

[20] J. Park, S. J. Koh, S. G. Kang, and D. Y. Kim, "Multicast delivery based on unicast and subnet multicast," *IEEE Communications Letters*, vol. 5, no. 4, pp. 181–183, 2001.

[21] K.-L. Cheng, K.-W. Cheuk, and S.-H. Gary Chan, "Implementation and performance measurement of an island multicast protocol," in *Proceedings of IEEE International Conference on Communications (ICC '05)*, vol. 2, pp. 1299–1303, Seoul, Korea, May 2005.

[22] K.-W. R. Cheuk, S.-H. Gary Chan, and J. Y.-B. Lee, "Island multicast: the combination of IP multicast with application-level multicast," in *Proceedings of IEEE International Conference on Communications (ICC '04)*, vol. 3, pp. 1441–1445, Paris, France, June 2004.

[23] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: a framework for delivering multicast to end users," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, vol. 3, pp. 1366–1375, New York, NY, USA, June 2002.

[24] X. Jin, K.-L. Cheng, and S.-H. Gary Chan, "SIM: scalable island multicast for peer-to-peer media streaming," in *Proceedings of IEEE International Conference on Multimedia & Expo (ICME '06)*, pp. 913–916, Toronto, Canada, July 2006.

[25] V. K. Goyal, "Multiple description coding: compression meets the network," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 74–93, 2001.

[26] K.-F. S. Wong, S.-H. Gary Chan, W.-C. Wong, Q. Zhang, W.-W. Zhu, and Y.-Q. Zhang, "Lateral error recovery for application-level multicast," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, vol. 4, pp. 2708–2718, Hong Kong, March 2004.

[27] W.-P. K. Yiu, K.-F. S. Wong, S.-H. Gary Chan, et al., "Lateral error recovery for media streaming in application-level multicast," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 219–232, 2006.

[28] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of the 15th Annual Joint Conference of the IEEE Computer and Communications Societies. (INFOCOM '96)*, vol. 2, pp. 594–602, San Francisco, Calif, USA, March 1996.