Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 51 (2007) 31–42
Computer Networks
www.elsevier.com/locate/comnet

Parallel overlays for high data-rate multicast data transfer

K.K. To, Jack Y.B. Lee *

Department of Information Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong

Received 17 September 2005; received in revised form 11 April 2006; accepted 25 April 2006
Available online 19 May 2006

Responsible Editor: G. Karlsson

Abstract

Recently a number of application-layer multicast (ALM) protocols have been proposed as a promising alternative to deploying multicast services in the unicast-only Internet. Current ALM protocols work very well for low data-rate applications but can suffer from link-level load imbalance, and consequently link congestions, when applied to high data-rate applications. This work addresses this problem by extending the well-known NICE protocol to use multiple parallel overlays in the same ALM session to spread the data traffic across more available network links, and thus leading to significantly improved performance. Extensive simulation results show that the proposed protocol can support three times the data-rate compared to NICE and yet can reduce the end-to-end data delivery delay by more than 50%.

Keywords: Application layer multicast; Parallel overlay network; Video streaming; High-data rate transfer; Tree-based topology

1. Introduction

The rapid deployment of broadband residential networks around the world has opened up a new channel for mass-media content distribution. Many of the traditional TV contents and many new video contents are now being regularly distributed and streamed over IP networks.

In mass-media content distribution, the same content is to be distributed to and received by a large number of receivers. This is precisely the type of network applications that can benefit from network multicast, such as IP multicast [1]. However, except for regional, semi-private broadband networks operated by a single service provider, the Internet at large unfortunately does not support native network multicast.

In response to this challenge, researchers have resorted to building overlay networks to implement multicast at the application layer – application-layer multicast (ALM). The principle of ALM is to route and forward multicast data using software running in hosts. The multicast data are tunneled through the underlying Internet using unicast transmission, and the participating hosts will replicate and forward these multicast data to other hosts in the overlay network until the messages reach the destined receivers.

Over the years, numerous ALM protocols have been proposed and deployed [2–11]. These protocols

* Corresponding author. Tel.: +852 26098377; fax: +852 26035032.
E-mail address: yblee@ie.cuhk.edu.hk (J.Y.B. Lee).

1389-1286/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2006.04.009

typically construct the overlay network based on some knowledge of the underlying network (e.g., delay or distance of the hosts from one another) [7–11]. Some also monitor the overlay network continuously to dynamically reconfigure the overlay network when network conditions change.

One such successful ALM protocols is the NICE protocol proposed by Banerjee et al. [9]. NICE is a tree-based ALM protocol where peers are arranged hierarchically such that every peer receives data from its parent or siblings and forwards the data to its children and siblings. This protocol has been shown to work well in many applications and networks due to its proximity-aware feature and its capability to dynamically adapt the overlay network topology to the changing network conditions.

Our work is motivated by the application of the NICE protocol to high data-rate applications, such as video content distribution. Specifically, our simulations show that although the NICE protocol works well for low to medium data-rate applications, it may run into performance bottlenecks when the data-rate approaches the capacity of some of the links in the underlying network. In this case the congested link will experience significantly increased delay and packet loss.

In theory, the protocol will be able to detect the link quality degradation and trigger a new round of overlay topology rearrangement to adapt to the detected congestion. However, in practice this may not always be able to resolve the problem as the topology rearrangement merely selects another link for data transport and the new link may then become the new point of congestion. This will trigger another topology rearrangement and so forth, and the overlay network will then become unstable.

This work tackles this problem by developing a parallelized version of the NICE protocol – P-NICE. The principle is to separate the data stream into multiple sub-streams, and then send each sub-stream over an independent multicast overlay. The sub-streams are then resequenced at the receiver before passing on to the application. As the sub-stream data-rate is much lower than the video bit-rate, the aforementioned link-congestion problem is significantly reduced. Moreover, different overlays can route data over disjoint links to fully utilize the available network capacity, and high-capacity links can also be fully utilized by routing multiple overlays through them.

Our extensive simulations show that P-NICE can increase the useable data-rate by over 300% (with 5 overlays) when compared to the original NICE protocol. More surprisingly, the need to resequence the sub-stream data at the receiving peer does *not* increase the end-to-end data delivery delay – the delay is in fact reduced by more than 50% when compared to NICE.

One tradeoff to the improved useable data-rate is the control overheads in managing the multiple overlays. To tackle this problem we develop a new algorithm based on exponentially weighted moving average to reduce the number and frequency of topology rearrangements, without adverse performance impact.

In the rest of the paper we first review some related work in Section 2, and then present the P-NICE protocol in Section 3. Using extensive simulations we evaluate the P-NICE protocol and compared it to the original NICE protocol in Section 4. We summarize the paper and discuss some future work in Section 5.

2. Background and related work

In the past decade researchers have proposed many successful ALM protocols. They can be classified into mesh-based and tree-based overlays in terms of the topology of the overlay network.

2.1. Single-overlay approaches

For mesh-based overlays, Chu et al. proposed the Narada protocol [10] which exploits the topological proximity of peers to reduce network resources consumed in multicast data delivery. Initially, peers are connected randomly to form a mesh-based overlay. The distances between peers are then estimated from round-trip-time (RTT) measures through periodic probing. Peers which are close together topologically are then connected and made closer in the overlay network. During operation, it progressively adds good paths to and removes bad paths from the mesh. However, the control overheads of Narada is in the order of $O(N^2)$ [10] and so it is only suitable for overlays with small number of peers.

In another work Pendarakis et al. [12] proposed an ALM middleware called ALMI to facilitate the porting of native multicast applications to the ALM environment. Similar to Narada, ALMI is also designed for small-size multicast group (tens of members) using the many-to-many communication pattern (e.g., voice or video conferencing).

However, in mass-media content distribution the data distribution topology is usually one-to-many and the size of the multicast group will be significantly larger. Obviously Narada and ALMI are not designed for this type of applications and thus will suffer from scalability issues.

In contrast, tree-based overlays have the potential to achieve significantly lower control overheads, particularly when the multicast group is large. For example, Banerjee et al. developed the NICE protocol [9] that builds tree-based rather than mesh-based overlay networks. NICE uses RTT to measure the topological proximity of peers and then arranges them into a hierarchical topology to localize the control messages. This significantly enhances the NICE protocol's scalability to large multicast groups and this is the primary reason why we choose to extend NICE instead of Narada or ALMI.

Nevertheless, NICE was initially designed for relatively low data-rate network applications. Therefore the network traffic generated by the ALM protocols is assumed to be insignificant when compared to the network link capacities. This assumption will not hold for high data-rate applications, even when the application data-rate reaches only 20% of the stub-to-stub link bandwidth (see Section 4.1).

This phenomenon is primarily due to the way multicast data are delivered over the overlay network. In particular, the tree-based overlays may not utilize all available network links due to the single route from the sender to a receiver. Consequently, when the application data-rate approaches that of the link capacity, congestion will occur, leading to increased delay and loss. Even though the ALM protocol can detect and adapt to link quality degradations, such adaptation was designed primarily for recovering from peer and link failures rather than *self-induced* congestions. Thus the relocation of the congested overlay path to another network link will merely relocate the point of congestion.

2.2. Multi-overlay approaches

One solution to the previously-mentioned problem is to match the data-rate to the available link capacities using multiple overlays. For example, Wang and Chan [7] proposed a centralized architecture to build a high bandwidth overlay. Multiple trees are built on top of the peers and the trees reach different subset of peers. Construction of the trees starts with the topology discovery process. Peers use *traceroute* to discover the physical paths between peers and then initialize a series of short TCP file transfers to estimate the available bandwidth. The information is then reported back to the central server which then constructs the distribution trees one by one. In each round, a tree is built so that it uses up all the residual bandwidth of at least one peer. The peers without any residual bandwidth are left out in the subsequent rounds. As a result, trees built in later rounds would reach progressively fewer peers and special data coding methods, like erasure coding and multiple descriptions coding (MDC) [13], must be used to enable the peers to decode the received data correctly.

In another study, Zhu et al. proposed the oEvolve protocol [8] that employed a decentralized approach to build multiple layers of trees in a distributed manner. Again as not all peers are reached in every tree, different peer will receive data at a different rate. Both studies [7,8] provide effective solutions to maximize the data-rate for individual peers as long as the data being distributed are partially decodable (e.g., video encoded in MDC [13] or fine-grained scalable video coding [14]).

For applications which do not employ partially-decodable data, Castro et al. proposed a SplitStream protocol [11] that also builds overlay network from multiple trees, with all trees reaching all receiving peers. The design goal of SplitStream is to balance the peers' forwarding load, i.e., the amount of data forwarded. It achieves this by building *interior-node-disjoint* trees to spread the forwarding load across multiple peers.

Node-level load balancing works well when the node-level bandwidth is the same or smaller than the link-level bandwidth. However today's peers are often PCs or workstations which easily forward tens, if not hundreds, of Mbps of data. Moreover, it is very common for many hosts to share an Internet access gateway that has significantly lower bandwidth (e.g., 1.5 Mbps for ADSL) than the individual peer's bandwidth (e.g., 10 or even 100 Mbps). In these cases the interior-node-disjoint approach will become sub-optimal.

Fig. 1 illustrates these scenarios using a simple network configuration. We first consider the scenario where the node-level bandwidth constraints match the link-level bandwidth constraints. In Fig. 1a there are one source peer (peer A) and three receiving peers (peers B, C, and D). Assuming the

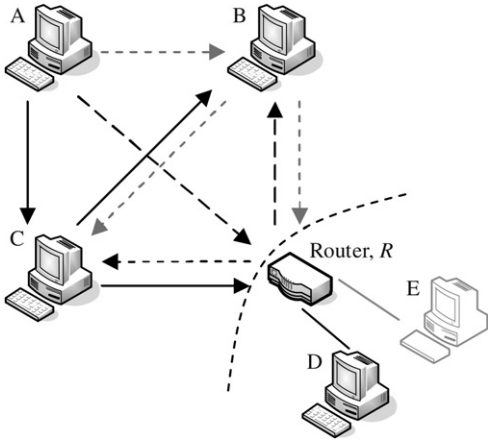

Fig. 1a. A case for SplitStream protocol when peers' node-level constraints match with link-level constraints.

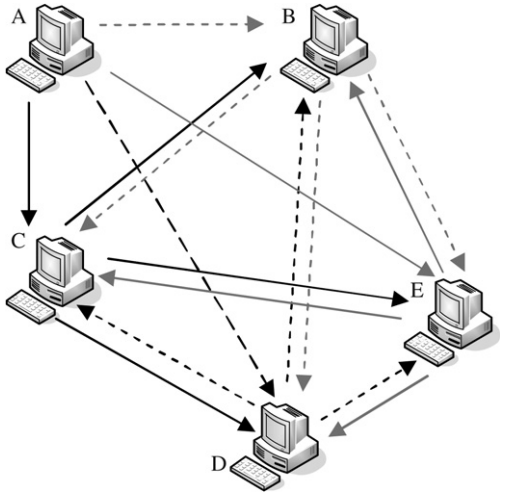

Fig. 1b. A case for SplitStream protocol when peers' node-level constraints do not match with link-level constraints.

peers all have the same node-level bandwidth constraint, the overlays will be constructed as shown in Fig. 1a. The data stream is divided into three overlays, represented by solid, dotted, and light-dotted lines respectively. Each arrow represents one unit of data flow and the total throughput of the multicast stream is 3 units of data flow.

Next we introduce a new peer E sharing the same access router R with peer D as shown in Fig. 1b. We assume the router R has link-level incoming (i.e., traffic going from outside to Peer D and/or E) bandwidth constraint of 3 units of data flow. After the new peer E, with identical node-level constraint as peer D, joined the ALM session, the interior-node-disjoint algorithm will have a total of four receivers, all of the same node-level bandwidth constraint. Consequently it will divide the data stream into four sub-streams and assign one of them to peers B, C, D, and E respectively for forwarding to other peers.

However as peer D and E share the same access router R with a link-level bandwidth constraint of 3 units, each of them can only utilize half the link bandwidth (i.e., 1.5 units) to receive sub-stream data from peer A and two other overlays (peer B and C), resulting in a maximum rate of $1.5/3 = 0.5$ unit of data flow per sub-stream. As a result the maximum achievable throughput will be limited to $0.5 \times 4 = 2$ units of data flow.

This scenario is clearly sub-optimal as we can increase the achievable throughput to 3 units simply by forwarding all sub-stream data via peer D to peer E, and vice versa. Nevertheless this is not possible in SplitStream due to the interior-node-disjoint constraint, which limits each peer to forward data for one and only one overlay.

In comparison, the P-NICE protocol investigated in this work achieves load balance at the link level rather than at the node level. On one hand, P-NICE allows multiple overlays to route data through the same node, if that node has the extra capacity to support them. This enables P-NICE to explore and fully exploit the network capacities.

On the other hand, unlike node-level bandwidth constraints, which are specified by the end user or preconfigured, link-level bandwidth often varies from time to time in the presence of competing traffics. Using the NICE protocol's continuous delay probing mechanism the P-NICE protocol can dynamically adapt to these link-level bandwidth variations to cope with congestions as well as to explore newly available link bandwidth. Our simulation results show that this works surprisingly well and can achieve balanced load across all trees even without any coordination between them.

2.3. Optimization approach

In a recent work, Cui et al. [15] approached the ALM problem from an optimization angle. Specifically, they developed optimization algorithms to compute the max-flow and max-concurrent-flow of transporting data over multiple overlay networks in a given physical network. Their work provided a theoretical understanding of the interactions and behavior of multiple overlay networks and also addressed other important issues such as fairness. However, the optimization methods assume knowl-

edge of the network topology and complete link capacity information, which may not be available in practice. NICE and P-NICE, in comparison, are designed to explore (through periodic delay measurements) and adapt (through local tree reorganizations) to the unknown physical network without any prior information of the network.

An interesting possibility is to apply Cui et al.'s optimization methods by replacing the link weight in their model with measured delay to reflect the current link utilization. There are, however, two potential problems. First, as the delays are measured at runtime, possibly in the presence of competing traffics, their measured value will inevitably vary from time to time. Consequently, subsequent runs of the optimization methods may produce different overlay configurations as a result of the changing delays. Unless the new overlay configuration closely resemble the existing one, extensive reconfigurations of (all) the overlays will be needed, leading to significant overheads as well as service degradation. Second, unlike link weight, which is invariant with respect to the actual overlay configurations generated, the node-to-node delays are dependent on the actual overlay configuration adopted. Thus after each optimization is completed the new overlay configuration will very likely lead to changes in the set of measured delays, and so render the optimized overlay configuration no longer optimal according to the optimization model. Not only that this will likely result in continuous reconfiguration of the overlays every time the optimization methods are invoked, the generated overlay configuration is in fact no longer optimal even in principle as the link weights are no longer independent from the optimization results.

3. P-NICE

We first review the original NICE protocol in Section 3.1 and then present two modifications in Sections 3.2 and 3.3 to extend the NICE protocol to high data-rate applications.

3.1. The original NICE protocol

NICE [9] is an ALM protocol designed to achieve low control overhead and to exploit peers' proximity in building the hierarchical overlay network. Peers in NICE obtain proximity information through measuring the RTT to other peers. Peers that are close together are then arranged into a number of clusters, with each cluster consisting of between k and $3k - 1$ peers, where k is an adjustable parameter. In addition to the clusters, peers are also arranged hierarchically with all peers belonging to layer 0. For each cluster at layer L, where $L \geqslant 0$, the graph theoretic center of the cluster is selected as the cluster leader to join layer $L + 1$, thus forming a cluster with other peers at layer $L + 1$. The process continues until there is only one peer at the uppermost layer.

Cluster leaders are responsible for forwarding and receiving data to and from their clusters respectively. On receiving data packets from its Lth layer cluster member, the leader forwards them to all members of the clusters it belongs to, except the members in its Lth layer cluster. As the leader is the topological center of its cluster and data is routed from leaders to leaders, NICE can achieve data delivery paths with low stretch (i.e., the averaged number of hops to deliver a data from source to destination).

In NICE there is a bootstrap host called the Rendezvous Point (RP) which is known to all peers before they join the multicast session. The RP maintains a list (not necessarily complete) of peers participating in NICE. When a new peer A joins NICE, it first contacts the RP to obtain a list of peers that are present in the highest layer, say Lth layer. Peer A than finds the peer, say peer B, in the list that is closest, based on RTT measurements, to itself and asks peer B to obtain its list of $(L - 1)$th layer cluster members. The process continues until peer A finds a peer in the 0th layer cluster, and peer A completes the process by joining that cluster. Peer A periodically conducts RTT measurements between itself and the cluster leader's siblings in its highest layer, say Lth layer. If a closer peer C, other than the cluster leader, is found, then peer A will leave the current highest layer cluster and join the Lth layer cluster of which peer C is the leader. This allows NICE to improve its topology in response to the changing network conditions.

3.2. Parallel overlay architecture

It is easy to see that in the NICE protocol data delivery from a source to a destination will go through a single path in the network. To take advantages of path diversity and to spread the traffic across more network links, we propose to build not one, but multiple overlay networks for data distribution.

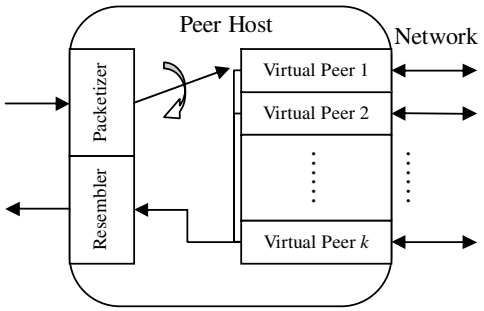

Fig. 2. Architecture of P-NICE, showing the k virtual peers.

Fig. 2 depicts the architecture of the proposed P-NICE protocol. In each ALM session, k overlays are built independently using the NICE protocol. Each peer is then sub-divided into k virtual peers (VP), with each virtual peer joining a different NICE overlay. To transmit data, the sending peer first packetizes data into packets of size P_k bytes and then distributes them to the virtual peers in a round-robin manner. The virtual peers in turn send them over the k NICE overlays independently. To receive data, the virtual peers of the receiving peer first receives the packets from the overlays, and then resequences them in the proper order before passing them to the application.

Compared to the single-overlay NICE protocol, this parallel-overlay P-NICE protocol allows data packets of the same ALM session to travel through different paths in the network. Although the k overlays may start out using similar network paths, the serialization delays at the lower network layers and the increased delay and loss due to the self-induced congestion will affect the measured RTT information of different peers. The k VPs then independently adapt their tree topologies base on continuous RTT measurements. As the measured data of different VPs, even if they reside within the same physical peer, will likely to be different, only some of the VPs will be triggered to adapt and select alternative paths to deliver their data sub-stream to relieve the detected link congestion. In time these k overlays will each settle on a particular set of delivery paths (not necessary disjoint) so as to equalize the load across the network links.

3.3. Control overheads

The main drawback of constructing k overlays for the same ALM session is the increased control overheads. In the original NICE protocol, peers continue to probe each other periodically to monitor any changes in network conditions. When the measured RTT changes, it will trigger a rearrangement of the overlay topology in an attempt to improve performance. Worst, when one of the k overlays in P-NICE rearranges its overlay topology, the other overlays sharing the same links will also experience changes in their RTT measurements. This will trigger even more overlays to start rearranging their topologies, thus forming a positive feedback loop.

To tackle this problem, we need to reduce the sensitivity of the triggering mechanism to short-term RTT variations while at the same time maintaining its ability to adapt to longer-term changes in the network. We propose to smooth out the RTT estimates using an exponentially weighted moving average (EWMA) algorithm similar to the one adopted in TCP for this purpose [16].

Specifically, the current estimated RTT is computed from

$$\overline{\mathrm{RTT}}_j = (1-\alpha)\cdot\overline{\mathrm{RTT}}_j\cdot + \alpha\cdot \mathrm{RTT}_j,$$
$$\Delta\mathrm{RTT}_j = (1-\beta)\cdot\Delta\mathrm{RTT}_j + \beta\cdot(\mathrm{RTT}_j - \overline{\mathrm{RTT}}_j),$$

where RTT_j is the measured RTT of peer j, with α and β equal to 0.125 and 0.25 respectively [16]. The system will then use $\overline{\mathrm{RTT}}_j$ instead of RTT_j in the triggering mechanism. To further filter out transient variations in the RTT, the system will not trigger as long as the new RTT is within

$$\overline{\mathrm{RTT}}_j - 4\cdot\Delta\mathrm{RTT}_j < \mathrm{RTT}_j < \overline{\mathrm{RTT}}_j + 4\cdot\Delta\mathrm{RTT}_j.$$

4. Performance evaluations

In this section, we present simulation results to evaluate the proposed P-NICE protocol and compare it to the original NICE protocol. A 10000-node network is generated using GT-ITM [17]. The transit-to-transit, transit-to-stub, and stub-to-stub link bandwidth is assigned as 100 Mbps, 8 Mbps and 4 Mbps, respectively. In each node (i.e., router), separate queues are built for each link and each queue has 64 KB buffer. The data stream is packetized into packets of 1 KB each. On top of the physical network, 128 peers are randomly attached to the nodes. Table 1 summarizes the default system parameters used in the simulations.

The peers join the ALM session randomly during the first 300 s and the data stream is started at 1300 s. Note that performance is measured during

Table 1
Default System parameters

Parameter	Value
Physical nodes count	10,000
Router buffer (per link)	64 kB
Transit-transit BW	100 Mbps
Transit-stub BW	8 Mbps
Stub-stub BW	4 Mbps
Data packet size, P_k	1 kB
Peers count	128
Data stream bitrate	800 kbps
Overlays in an ALM session, k	5
Simulation time	4000 s
Simulation trials	5

the entire simulation duration rather than only after the system has reached steady state. This captured the performance of the ALM protocols over its whole life cycle and thus allows us to study their temporal behavior. In fact the ALM protocols, even long after the ALM session had started, will continue to refine the data distribution topology in response to RTT measurement updates.

4.1. End-to-end packet delivery ratio

Fig. 3 shows the end-to-end packet delivery ratio versus different data rates, where the vertical bars represent the 95% confidence intervals. Delivery ratio is defined as the ratio between total number of data packets received by all peers and the number of data packet expected to be received by all peers if all deliveries are successful. From the simulation result, it is clear that with only a single overlay the NICE protocol can achieve a data-rate up to only 400 kbps with approximately 80% delivery ratio. However, by using P-NICE with 5 overlays, the achievable throughput increases significantly to 1200 kbps with 90% delivery ratio, which is three times higher than NICE. This result clearly shows that it is difficult to fully utilize the available network resources using only a single overlay network to deliver multicast data.

A second observation is that the control overhead reduction algorithm described in Section 2.3 does not result in any significant performance degradation. In fact the delivery ratio is slightly higher at high data rates (e.g., at 1800 kbps) compared to using the simple RTT-based algorithm in NICE. This is because topology rearrangement itself also causes data loss – during the time when the paths are being rearranged. Thus by reducing the number and frequency of topology rearrangements, which is more significant when the delivery ratio is low, the delivery ratio can be improved.

4.2. Utilization of network links

To pinpoint the source of the improvements in the previous section, we need to study the utilization of the network at the link layer. Fig. 4 plots the link-layer packet loss rate versus data stream bit-rate, where the vertical bars represent the 95% confidence intervals. We observe that NICE experiences significant packet loss when the data-rate exceeds 500 kbps. With P-NICE the loss rate is less than 1% even at a much higher data-rate of 1200 kbps. This suggests that P-NICE can effectively spread the data traffic across more network links to avoid congesting a few links.

This is evident in Fig. 5a, which shows the distribution of maximum utilization of each link

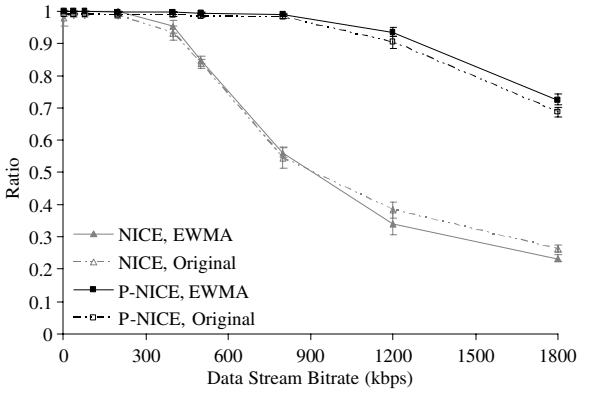

Fig. 3. End-to-end packet delivery ratio versus data stream bit-rate.

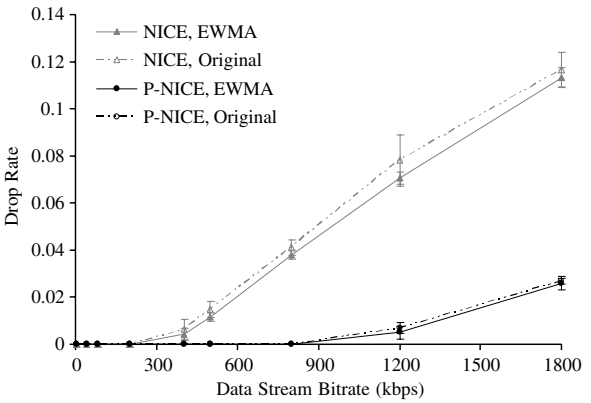

Fig. 4. Link-layer packet drop rate versus data stream bit-rate.

throughout the simulation. It is easy to see that under NICE there is a substantial proportion of links operating in near full capacity (e.g., about 3% are running at 100% utilization). Obviously these links are congested, leading to significant packet loss. By contrast, under P-NICE most of the links are operated at much lower utilization, with only a few (0.4%) reaching full utilization.

Note that results in Fig. 5a merely show how many of the links, but not how *often* the links are being operated at full utilization. To investigate the latter we plot in Fig. 5b the distribution of maximum link utilization across time. The results show that in NICE over 16% of the time one or more links are being operated at full capacity. By contrast, in P-NICE only 0.05% of the time there exist at least one fully-utilized links. This shows that link utilization under P-NICE is far more balanced than NICE, thus resulting in the much improved delivery ratio at higher data-rates.

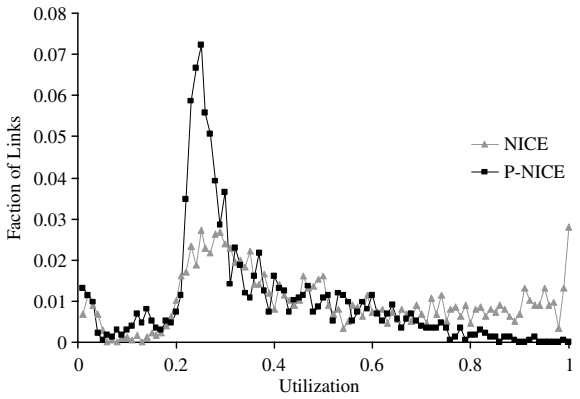

Fig. 5a. Distribution of maximum link utilization.

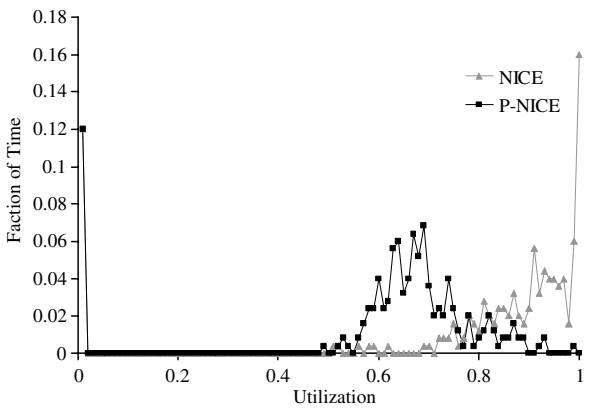

Fig. 5b. Distribution of maximum link utilization across time.

4.3. Effect of number of overlays

The previous results for P-NICE are simulated using 5 overlays. To investigate the effect of the number of overlays, we repeated the same simulation for number of overlays ranging from 1 to 10.

Figs. 6 and 7 plot respectively the average packet delivery ratio and the packet loss rate versus the number of overlays employed. We plotted two sets of curves for 1 and 5 sources sending out multicast data in the ALM session. We can observe that the number of sources have negligible impact on the performance while increasing the number of overlays improves both the delivery ratio and packet loss rate as expected. In this particular setup we only need 5 overlays to bring the delivery ratio to nearly 100%.

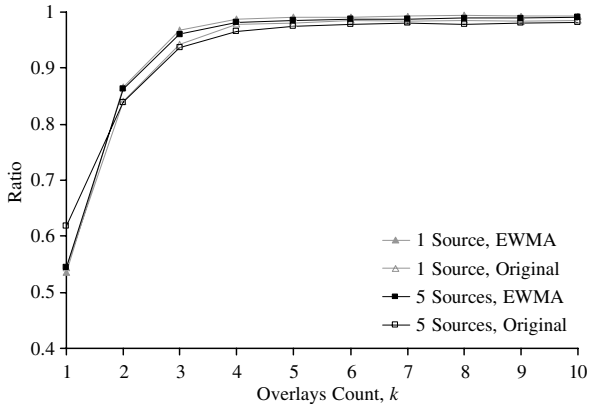

Fig. 6. Distribution of end-to-end packet delivery ratio for different number of overlays.

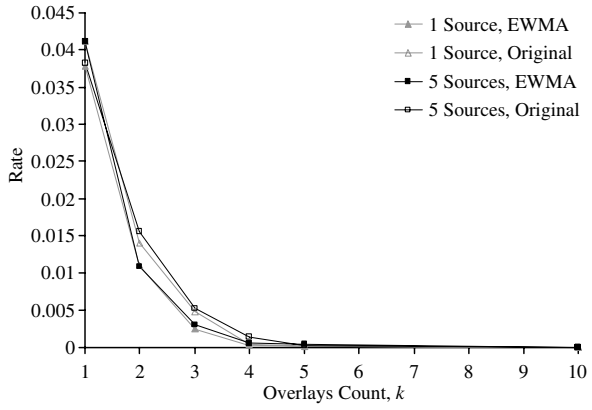

Fig. 7. Link-layer packet drop rate for different number of overlays.

4.4. End-to-end data delivery delay

With multiple overlays for data delivery, one would expect the receiver to experience longer data delivery delay due to the need for resequencing the data received from different overlays. We investigate this issue in Fig. 8 by plotting the distribution of end-to-end data delivery delay. Surprisingly, P-NICE in fact achieves significantly lower data delivery delay than NICE. In comparison, distribution of the data delivery delay under NICE appears to be bi-modal. This is because some links are heavily utilized, thus leading to higher queueing delay. Receiving peers that are *up-stream* of the congested links (represented by the first peak in Fig. 8) are not affected by the queueing delay and they experience much shorter data delivery delay than peers *down-stream* of the congested links (represented by the second peak in Fig. 8).

4.5. Load balance of overlays

It is worth noting that in P-NICE the overlays are all operated independently of each other. Nevertheless the overlays in time automatically spread themselves across the physical network to balance out the link utilizations. To see the performance of P-NICE in the start-up phase we plot in Fig. 9, the end-to-end delivery ratio versus simulation time. In this simulation we started the data stream at 300 s immediately after all the peers have joined the multicast session. The results clearly show that P-NICE can ramp up to full performance in a very short time and remain stable thereafter. By contrast, the performance of NICE is significantly lower and far more erratic.

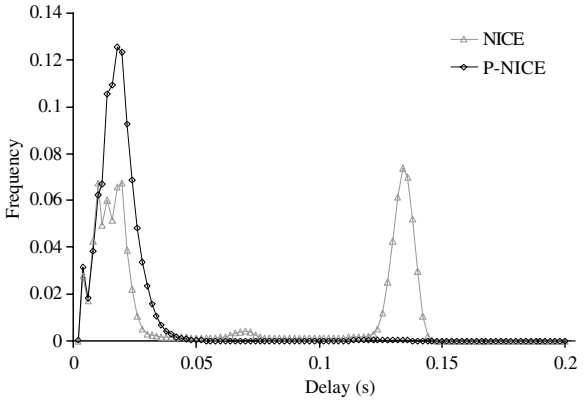

Fig. 8. Distribution of end-to-end packet delivery delay.

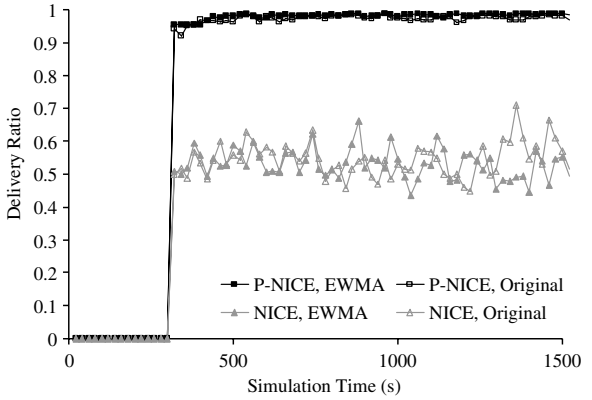

Fig. 9. End-to-end packet delivery ration over time.

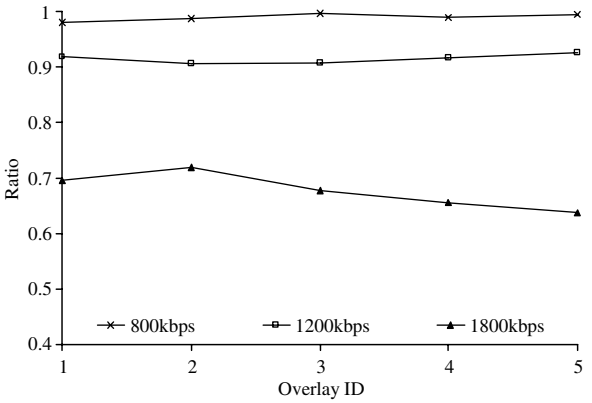

Fig. 10a. Packet delivery ratio for individual overlays within the same ALM session ($k = 5$).

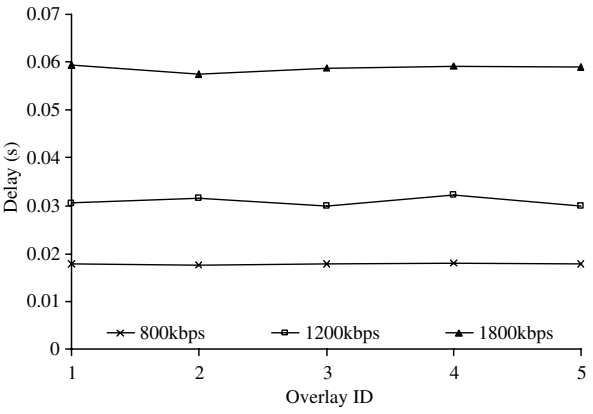

Fig. 10b. Packet delivery delay for individual overlays within the same ALM session ($k = 5$).

In Fig. 10 we further investigate the performance of individual overlays. We observe that both delivery ratio (Fig. 10a) and data delivery delay

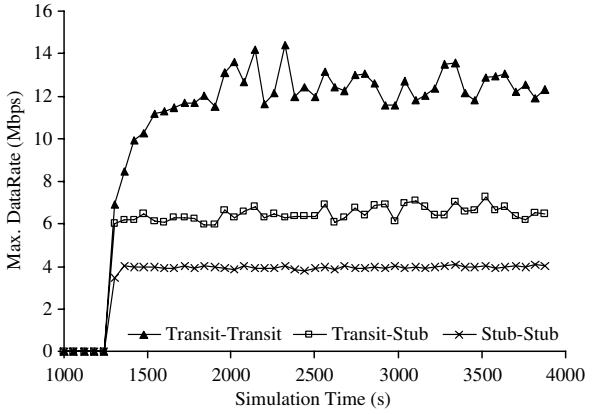

Fig. 11. Maximum data-rate of different type of physical links across time.

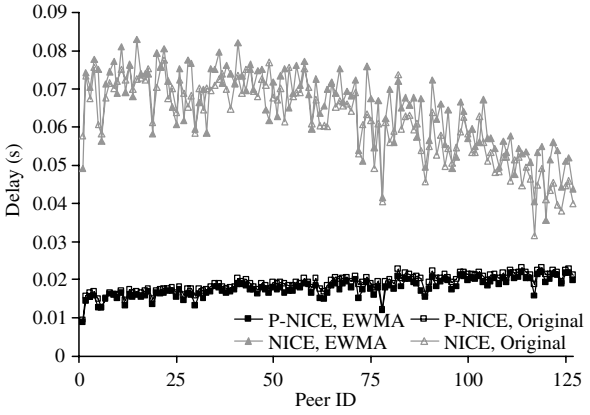

Fig. 12a. Distribution of end-to-end packet delivery delay for different peers within the same ALM session.

(Fig. 10b) are very similar for all overlays. This surprising result can be explained by comparing the link utilization of different types of links. Specifically, there are three different types of links in our simulation, namely transit-to-transit links (high bandwidth), transit-to-stub links (medium bandwidth), and stub-to-stub links (low bandwidth).

Fig. 11 shows the maximum data-rate of each of the link types across the simulation time period. The results show that P-NICE, despite its round-robin data allocation across the multiple overlays, can still exploit links with higher bandwidth (i.e., transit-to-transit and transit-to-stub) by routing more overlays through these links. In contrast, the low bandwidth links are being kept from congestion by reducing the number of overlays passing through them.

In other words, P-NICE adapts the data flow to a link's capacity by routing an appropriate number of overlays through it, without requiring explicit unequal rate allocation at the overlay level.

4.6. Peers reception quality

Fig. 12a shows the reception quality for different peers within the same ALM session. For P-NICE, peers experienced data delivery delay from 0.01 s to 0.025 s while peers in NICE experienced a wider variation from 0.03 s to 0.08 s. This shows that the delay performance for different peers is more consistent when using P-NICE. The same is observed in the delivery ratio shown in Fig. 12b, with peers in NICE experiencing delivery ratio between 40% and 60% while peers in P-NICE all achieves more than 98% delivery ratio. Overall, P-NICE with multiple overlays can achieve far more consistent performance across all receiving peers.

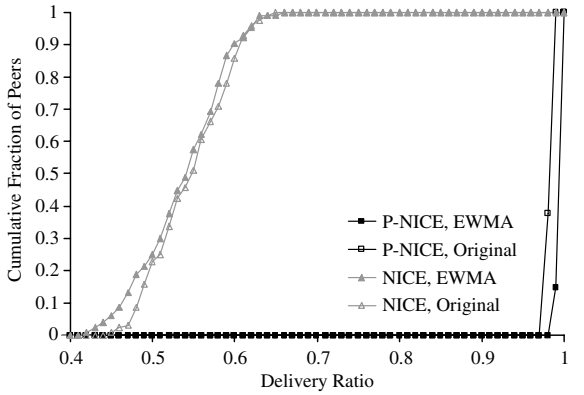

Fig. 12b. Cumulative distribution of end-to-end packet delivery ratio of peers.

4.7. Control overheads

Lastly, we investigate the control overheads generated by the ALM protocols. Fig. 13 plots the total control overheads of P-NICE (in megabytes) for different number of overlays, over the entire simulation period. The overhead of the original NICE protocol is about 40 MB while the overhead of P-NICE with 5 overlays is about 6 times more at 240 MB. However, we note that the delivery ratio of P-NICE and NICE differ significantly, with P-NICE achieving 98% and NICE achieving only 55%.

To compare the control overheads more fairly, we introduce a metric called *Normalized Overhead*, defined as the ratio between the total control over-

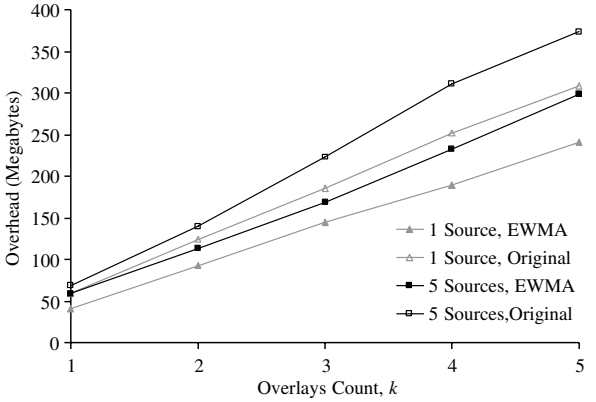

Fig. 13. Total overhead versus number of overlays.

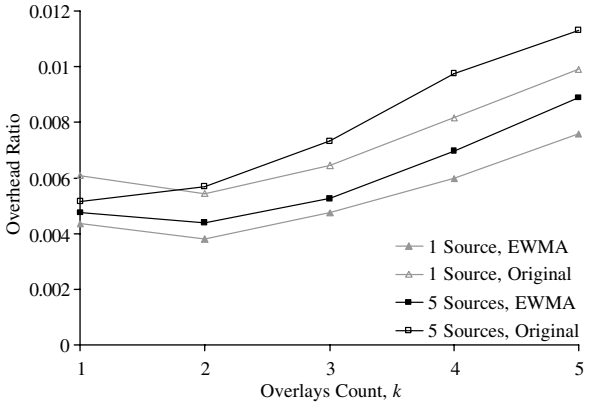

Fig. 14. Normalized overhead versus number of overlays.

heads (in bytes) over the total amount of data received by all peers. In other words, the Normalized Overhead represents the cost of control overhead per unit of data delivered.

Fig. 14 plots the normalized overhead for 1 to 5 overlays. The overhead for a peer is about 4–6 kbps when transporting a 800 kbps media stream. This is quite small (~0.7%) when compared to the data-rate of 800 kbps. Even for 5 overlays the normalized overheads can still be kept within 1% of the media stream data-rate and so the impact of RTT measurements on the data transfers will be negligible.

5. Summary and future work

In this work we extended the NICE protocol to use parallel overlays to better utilize the available network capacities to achieve a higher multicast data throughput. The principle behind P-NICE is the routing of application-layer multicast packets over multiple paths in the network. This multipath multicast routing scheme can be considered as an extension of multipath routing studied extensively in unicast applications [18].

To compensate for the increased control overheads, P-NICE employs a new algorithm based on EWMA to smooth out the measured RTT and to filter short-term random variations to reduce the number and frequency of topology rearrangements. The simulation results show that P-NICE substantially outperforms NICE in high-data-rate applications.

One technique adopted in P-NICE is to employ mechanisms to continuously measure the network conditions (in the forms of peer-to-peer delays) and then to react to network changes through topology reconfigurations. Instead of reacting to network condition changes, the ALM protocol could also actively optimize its performance through, for example, proactively probing for available network bandwidth or proactively reallocating the data-rates allocated to the parallel overlays. However, this will likely increase the control overheads further and thus it will be desirable to explore new ways to cut down the control overheads, such as through sharing measurement information across the multiple overlays,[1] or taking one step further, splitting the measurement tasks between the multiple overlays to further reduce control overheads.

Finally, as the number of overlays has a significant impact on the achievable throughput, it will be highly desirable to develop intelligent ways to control and optimize the number of overlays to deploy, or even to dynamically adapt the number of overlays in response to changing network conditions. Further work is warranted to investigate these and many other open problems in application layer multicast over parallel overlays.

Acknowledgements

The authors wish to express their gratitude to the anonymous reviewers and the associate editor for their constructive comments and insightful suggestions in improving this paper. This work was funded in part by a Direct Grant, an Earmarked Grant (CUHK4211/03E) from the HKSAR Research Grant Council, and the UGC Area of Excellence in Information Technology Scheme (AoE/E-01/99).

[1] The authors wish to thank the anonymous reviewer for this idea.

References

[1] A. Hu, Video-on-demand broadcasting protocols: a comprehensive study, in: Proceedings of the IEEE INFOCOM 2001, Anchorage, AK, April 2001, pp. 508–517.

[2] T. Lavian, P. Wang, R. Durairaj, D. Hoang, F. Travostino, Edge device multi-unicasting for video streaming, in: Proceedings of 10th International Conference on Telecommunications (ICT 2003), vol. 2, February 2003, pp. 1441–1447.

[3] D.K. Kim, K.I. Kim, I.S. Hwang, S.H. Kim, Scalable and topology-aware application layer multicast architecture, in: Proceedings of the 6th International Conference on Advanced Communication Technology (ICACT 2004), vol. 1, February 2004, pp. 15–20.

[4] Y.H. Chu, S.G. Rao, S. Seshan, H. Zhang, Enabling conferencing applications on the internet using an overlay multicast architecture, in: Proceedings of ACM SIGCOMM, August 2001.

[5] Y. Cui, B. Li, K. Nahrstedt, oStream: asynchronous streaming multicast in application-layer overlay networks, IEEE Journal on Selected Areas in Communications 22 (1) (2004) 91–106.

[6] M. Kwon, and S. Fahmy, "Topology-Aware Overlay Networks for Group Communication," in: Proceedings of ACM NOSSDAV, May 2002.

[7] X. Jin, Y. Wang, S.H. Chan, Fast overlay tree based on efficient end-to-end measurements, in: Proceedings of IEEE ICC, May 2005.

[8] Y. Zhu, J. Guo, B. Li, OEvolve: toward evolutionary overlay topologies for high-bandwidth data dissemination, IEEE Journal on Selected Areas in Communications 22 (7) (2004).

[9] S. Banerjee, B. Bhattacharjee, C. Kommareddy, Scalable application layer multicast, in: Proceedings of ACM SIGCOMM, August 2002.

[10] Y. Chu, S.G. Rao, H. Zhang, A case for end system multicast, in: Proceedings of ACM Sigmetrics, June 2000.

[11] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, A. Singh, Splitstream: high-bandwidth multicast in cooperative environments, in: Proceedings of 19th ACM Symposium on Operating Systems Principles (SOSP), October 2003.

[12] D. Pendarakis, S. Shi, D. Verma, M. Waldvogel, ALmi: an application level multicast infrastructure, in: Proceedings of 3rd Usenix Symposium on Internet Technologies & Systems (USITS), March 2001.

[13] J.G. Apostolopoulos, Reliable video communication over lossy packet networks using multiple state encoding and path diversity, in: Proceedings of Visual Communications and Image Processing, January 2001.

[14] H.M. Radha, M. van der Schaar, Y.W. Chen, The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP, IEEE Transactions on Multimedia 3 (1) (2001) 53–68.

[15] Y. Cui, B. Li, K. Nahrstedt, On achieving optimized capacity utilization in application overlay networks with multiple competing sessions, in: Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), Barcelona, Spain, June 2004.

[16] V. Paxson, M. Allman, Computing TCP's retransmission timer, RFC 2988, November 2000.

[17] E. Zegura, K. Calvert, S. Bhattacharjee, How to model an Internetwork, in: Proceedings of IEEE INFOCOM '96, vol. 2, March 1996, pp. 594–602.

[18] E. Gustafsson, G. Karlsson, A literature survey on traffic dispersion, IEEE Network 11 (2) (1997) 28–36.

K.K. To received the B. Eng. and M.Phil. degrees in Information Engineering from the Chinese University of Hong Kong in 2003 and 2005 respectively. He was a researcher at the Multimedia Communications Laboratory in the same department and is now with the ASTRI research institute. His research focuses on multimedia communications, multicast, and video streaming systems.

Jack Y.B. Lee received the B.Eng. and Ph.D degrees in Information Engineering from the Chinese University of Hong Kong in 1993 and 1997 respectively. He participated in the research and development of video streaming systems from 1997 to 1998 where he and his team developed novel parallel video server architectures for building cost-effective, scalable and fault-tolerant video-on-demand systems. This work had resulted in numerous publications, two US Patents, and the technologies are subsequently transferred to a spin-off technology company for commercialization. He was a faculty member at the Department of Computer Science at the Hong Kong University of Science and Technology from 1998 to 1999, and in 1999 he joined the Department of Information Engineering at the Chinese University of Hong Kong to establish the Multimedia Communications Laboratory (http://www.mcl.ie.cuhk.edu.hk) to spearhead research in distributed multimedia systems, fault-tolerant systems, peer-to-peer systems, multicast communications, and Internet computing.