J. Vis. Commun. Image R. xxx (2009) xxx–xxx

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

CORS: A cooperative overlay routing service to enhance interactive multimedia communications

Li Tang [a,c,*], Zhen Chen [c,d], Hao Yin [b,c], Jun Li [c,d], Yanda Li [a,d]

[a] Department of Automation, Room 3-421, FIT Building, Tsinghua University, Beijing, China
[b] Department of Computer Science, Tsinghua University, Beijing, China
[c] Research Institute of Information Technology, Tsinghua University, Beijing, China
[d] Tsinghua National Laboratory for Information Science and Technology, Beijing, China

ARTICLE INFO

Article history:
Received 30 December 2008
Accepted 22 June 2009
Available online xxxx

Keywords:
Interactive multimedia communications
Overlay routing
Multi-path transmission service

ABSTRACT

As interactive multimedia communications are developing rapidly on the Internet, they present stringent challenges on end-to-end (E2E) performance. On the other hand, however, the Internet's architecture (IPv4) remains almost the same as it was originally designed for only data transmission purpose, and has experienced big hurdle to actualize QoS universally. This paper designs a cooperatively overlay routing service (CORS) aiming to overcome the performance limit inherent in the Internet's IP-layer routing service. The key idea of CORS is to efficiently compose a number of eligible application-layer paths with suitable relays in the overlay network. Besides the direct IP path, CORS can transfer data simultaneously through one or more application-layer paths to adaptively satisfy the data's application-specific requirements on E2E performance. Simulation results indicate the proposed schemes are scalable and effective. Practical experiments based on a prototype implemented on PlanetLab show that CORS is feasible to enhance the transmission reliability and the quality of multimedia communications.

1. Introduction

In recent years, audio and video techniques have been widely used to provide users with racy and lifelike experiences while they are interactively communicating on the Internet. Unlike conventional Internet applications such as Email and file transferring, the quality of interactive multimedia communications is highly dependent on end-to-end (E2E) performance. On the other hand, however, the Internet's current architecture (IPv4) almost remains the same as its original design based on the principles fitting data transmission applications. Therefore, despite the rapid increase of backbone and access bandwidth, it is still stringent for the current Internet infrastructure to support high-quality interactive multimedia communications.

1.1. Characteristics of interactive multimedia communications

A typical existing application of interactive multimedia communications generally consists of the following functional components. First, the sender's media equipment (such as camera and microphone) samples and converts the user's audio/video signals into digital data. Then, the application software encodes the data based on certain audio/video coding standard. The encoded multimedia data are usually transmitted with RTP/UDP protocol through the Internet. On receiving a packet, the receiver's application system stores the data into a jitter buffer to alleviate the packet disorder and delay variation caused by network dynamics. At last, the buffered data are picked out, decoded and played out as the same speed and in the same order as they were generated by the sender.

From the transport-layer point of view, interactive multimedia communications differ from conventional data transmission applications on several aspects. In data transmission applications, as the data received should be exactly the same as what have been sent, all packets are equally important in the sense that packets do not have to be transmitted in a specific order and every lost packet has to be retransmitted until they all successfully reach the destination. In contrast, in interactive multimedia communications, the significance of packets that carry multimedia data can be quite different from each other. Most audio/video coding standards generate two types of data: one type (such as an intra-frame) can provide useful information independently, while the other type (such as an inter-frame) has to rely on other data to be of real use. Obviously, the former type of data is more significant than the latter one, and thereby should be particularly protected during the transmission.

Moreover, the quality of interactive multimedia communications is delay critical, which makes the retransmission mechanisms no longer as effective as they are in data transmission applications.

* Corresponding author. Address: Department of Automation, Room 3-421, FIT Building, Tsinghua University, Beijing, China.
E-mail address: tangli03@mails.tsinghua.edu.cn (L. Tang).

1047-3203/$ - see front matter © 2009 Published by Elsevier Inc.
doi:10.1016/j.jvcir.2009.06.005

Because interactive multimedia communications can only generate and play out data in chronological order, the retransmitted packets are hardly able to arrive in time. Note that augmenting the jitter buffer on the receiver side is not as effective as in video streaming systems either, because its side effect on increasing E2E delay will severely harm the interactivity and user experience of communications.

Although RTP protocol has employed many E2E transport functions that are suitable for transmitting real-time multimedia data, it still provides no mechanism to ensure timely delivery or other QoS guarantees, but depends on the lower-layer service to do so. As a result, RTP packets that carry interactive multimedia data are actually transmitted through the same route by the Internet's IP-layer routing service, and will suffer from all the following limitations.

1.2. Limitations of IP routing and transmission service

In face of the characteristics of interactive multimedia communications, the current IP-layer routing and transmission service manifests the following limitations.

First, the current IP-layer routing is designed to provide 'best-effort' service without any guarantee on the E2E delay, packet loss, or usable bandwidth. Although deliberate retransmission mechanisms have enabled TCP protocol to provide reliable E2E data transmission service, it can hardly meet the delay requirement of interactive multimedia communications that demand one-way delay no larger than 400 ms for acceptable user experience [1]. Thus, most applications of interactive multimedia communications use RTP/UDP protocol and thus have to directly suffer from the Internet's unstable E2E performance.

Secondly, today's IP-layer routing service transfers all kinds of traffic through an identical E2E route, which is inflexible to satisfy the different requirements of diverse applications. In particular, the single route determined by the IP-layer routing service is often far from optimal to fit the particular demand of interactive multimedia communications, because the demand also depends on the multimedia coding schemes and the user's subjective experiences. Worse still, the interests of autonomous system (AS) and BGP in scalability and policy enforcement hide too much topology and traffic information that is necessary for the IP-layer routing service to optimize E2E performance. A variety of studies have revealed that many of the current Internet's E2E routes are in fact significantly inflated [2–4]. Given that there are a diversity of routes across the Internet between two end-hosts, the IP-layer's single-path routing is inefficient to utilize the network resources.

Moreover, while many efforts have been made on supporting QoS on the IP layer, they are seldom widely deployed in practice due to some insuperable practical difficulties, such as upgrading the legacy infrastructure and requesting different ISPs to coordinate their operations. As a result, so far there is no scheme universally implemented on the IP layer to differentiate the data's transmission reliability or priority according to the data's significance. Although most multimedia coding schemes produce packets having different impacts to user's experiences, application developers, given the current IP-layer routing service, can only increase the reliability of significant data by duplicate transmission. Unfortunately, however, continuously transmitting multiple copies of data through the same route is inadvisable and helpless, because packets are often lost in a burst pattern on the Internet [5,6].

Last but not least, the availability of the current IP-layer routing service is still much lower than that of the public-switched telephone networks [7,8], and has to be improved before interactive multimedia communications on the Internet become as attractive as other commercial services.

1.3. Objectives of CORS design

Inspired by all above observations, we propose a cooperative overlay routing service named CORS to complement the current IP-layer routing service to better support the E2E performance requirement of interactive multimedia communications.

The key idea of CORS is to provide enhanced transport-layer mechanisms by efficiently discovering, constructing and utilizing a number of application-layer paths that are comprised of proper relays in the overlay network. To achieve this goal, CORS maintain a scalable overlay network, in which every participant end-host that starts a session of interactive multimedia communications is able to ask other suitable participant end-hosts to construct application-layer paths. Thereafter, besides the direct IP path, the multimedia data can also be simultaneously transmitted through one or multiple application-layer paths adaptively according to the data's significance and delay requirements.

This paper presents the design and implementation of CORS. In particular, we emphasize on presenting its design rationales, architecture and pivotal schemes to achieve high scalability and efficiency in discovering and utilizing a diversity of high-quality application-layer paths. These are the basis for developing novel path manipulation schemes, source and channel coding schemes and their incorporation (such topics are separately discussed in our other relevant papers [9]). CORS has been implemented as a prototype and deployed on PlanetLab. Practical experiments show that CORS is effective to enhance data transmission reliability and the quality of multimedia communications.

The rest of this paper is organized as follows. Section 2 introduces the design rationales and architecture of CORS. Section 3 elaborates the main algorithms enabling CORS to support a large scale overlay network and to efficiently discover and utilize high-quality application-layer paths. Section 4 presents the evaluation experiments with a prototype implementation deployed on PlanetLab. Section 5 compares CORS with existing overlay routing schemes, and finally Section 6 concludes the paper.

2. Design rationales

To achieve its objectives, the design of CORS is confronted with three main challenges. First, CORS needs to be scalable enough to support a large number of participants, which is important to ensure the existence of suitable relays that can construct high-quality application-layer paths for a diversity of communication sessions. Secondly, CORS needs an efficient method that can discover suitable relays for a given session without triggering too much measurement traffic and overhead. Last, a dispatch algorithm is required to coordinate the direct IP path and a number of application-layer paths with different performance properties to properly transmit the multimedia data according to their requirements.

In this section, we first present an overall picture of CORS, including its architecture and functional modules, and then we explain the rationales why CORS is designed in its current way rather than adopting alternate schemes. Finally, we discuss the deployment, privacy and integration issues of CORS. The detailed algorithms will be introduced and evaluated in the next section.

2.1. Architecture

Conceptually, CORS can be considered as a middle-ware between the application software and the current Internet's transport-layer service. Fig. 1 shows the structure of a typical application of interactive multimedia communications utilizing CORS to enhance its data transmission.

Please cite this article in press as: L. Tang et al., CORS: A cooperative overlay routing service to enhance interactive multimedia communications, J. Vis. Commun. (2009), doi:10.1016/j.jvcir.2009.06.005

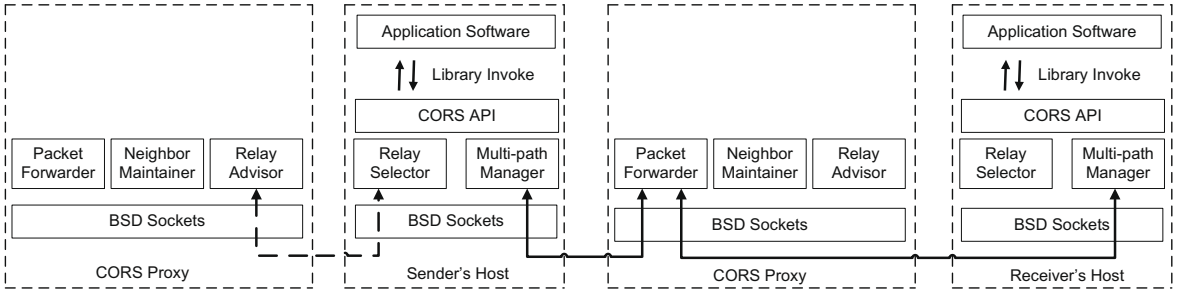


Fig. 1. Structure of a typical example application of interactive multimedia communications utilizing CORS to enhance its data transmission.

From the application software perspective, CORS provides a set of network programming APIs similar to the well-known BSD sockets. The main difference is that *CORS APIs* allow the application software to specify the data's significance and delay requirements. Invoked by the application, the sender's *CORS API* passes its inputs to the *multi-path manager* module, which then transmits the multimedia data with a proper combination of the IP path and a number of application-layer paths according to the data's specified requirements. Every relay along an application-layer path has to run a *packet forwarder* module as a daemon process to forward packets on application-layer for the sessions using this path. Finally, all packets successfully arriving will be received by the receiver's *multi-path manager* module, which will store the packets in a jitter buffer and hand in the reconstructed data to the receiver's application software through the corresponding *CORS API*.

As a functional distributed system, CORS also depends on the other three control-plane modules. For each participant, the *neighbor maintainer* module maintains a list of online neighbors, which allows CORS to form a large-scale overlay network in a scalable and distributed manner. The *relay advisor* and *relay selector* modules are used to efficiently find out high-quality application-layer paths. First, the *relay advisor* module roughly sifts out a set of candidate relays from a large number of online participants based on some heuristic metrics with rather low cost, and then the *relay selector* module performs active measurements to determine which candidate relays are able to compose application-layer paths that can offer capable E2E performance.

Next, we elaborate the design rationales of CORS, respectively, on control and data planes.

2.2. Control plane

Finding and composing an agreeable set of application-layer paths that are usable in interactive multimedia communications, control plane is the premise for all the other parts of CORS to serve their purposes. Stemming from the specific characteristics of interactive multimedia communications, the selected application-layer paths are expected to fulfill the following requirements:

Loop free: While simple at the first glance, this requirement actually deserves careful consideration, because CORS has to assure not to cause routing loops not only by a single path but also by any combination of multiple paths.

E2E delay: Considering that other components of the multimedia communication system also introduce delay (such as sampling duration and buffering duration), application-layer paths with smaller delay are preferred to achieve better user experience of interactivity.

Path diversity: Application-layer paths with less overlap with the IP path are preferred, because it can alleviate their performance correlation and reduce the probability for them to share the same physical bottleneck.

Although routing has been a well studied topic on the IP-layer for a long time, none of the popular IP-layer routing protocols is suitable for CORS to compose application-layer paths. To show this, we classify existing IP-layer routing protocols into three categories: distance-vector, link-state, and path-vector. The first two categories both rely on the shortest-path routing algorithm to avoid routing loops, which only works in the case of single-path routing. The third category does not work because it needs to maintain and propagate the whole path from the current place to the destination, and therefore requires a long time to converge after the network topology changes. Note that the change frequency of overlay network topology is much higher than that of the IP-layer topology, because the nodes in overlay networks are mostly played by end-hosts, which have much higher churn rate than common routers.

Moreover, all existing IP-layer routing protocols maintain the route in a proactive manner, which is not scalable to support large-scale overlay networks. Because thanks to the IP-layer routing service, almost every two nodes are directly connected in the overlay topology; maintaining routing states proactively in a topology with extremely high connectivity would require a huge amount of measurement traffic and cannot scale up. On the other hand, however, the scalability is important for CORS to cover diverse relays that can compose suitable application-layer paths to satisfy the requirements of E2E delay and path diversity.

Due to all above reasons, we design CORS to compose one-hop application-layer paths reactively. A one-hop application-layer path only involves a single relay and consists of two IP-layer paths: from the source to the relay and from the relay to the destination. Compared to using multiple-hop application-layer paths, using one-hop application-layer paths has the following advantages. First, it minimizes the number of relays along an application-layer path and thus minimizes the additional delay for relays to process and forward packets on the application-layer. Secondly, the impact of relays' churn on the reliability of application-layer paths is alleviated. Most importantly, using one-hop application-layer paths can be efficiently realized in a source-routing manner that enables the sender to completely control the construction and utilization of application-layer paths. In this way, not only the puzzles of avoiding routing loops and oscillation are inherently solved, but also the relays are saved from the complexities of propagating routing information and maintaining the packet-forwarding tables.

Despite the above advantages, one may concern that one-hop application-layer path could lead to performance limitation. However, previous work has shown that the best one-hop application-layer path is able to achieve performance gain close to the optimal

Please cite this article in press as: L. Tang et al., CORS: A cooperative overlay routing service to enhance interactive multimedia communications, J. Vis. Commun. (2009), doi:10.1016/j.jvcir.2009.06.005

application-layer path in the overlay network [10,4]. As there is a bijection between a one-hop application-layer path and its corresponding relay, in the rest of this paper we may interchangeably use these two concepts for ease of description.

By reactively, we mean that except those cached by previous sessions, CORS does not prepare or compose application-layer paths until the application software invokes *CORS API* to initiate a session of interactive multimedia communications. The session initialization triggers the *relay selector* module to perform the following operations: first, asking the *relay advisor* module for a collection of relay candidates; second, actively measuring the E2E performance and properties of the application-layer path composed by each of the recommended relay candidates; last, increasingly adding and updating the collection of agreeable application-layer paths used by the *multi-path manager* module. It is important to note that the above three operations are executed iteratively and continuingly until the whole session of interactive communications is terminated.

As it is cost prohibitive to measure the E2E performance of a large number of application-layer paths, one difficult challenge facing CORS is how to reduce the number yet increase the quality and diversity of the paths composed by those relay candidates that are recommended by the *relay advisor* module. For the sake of path diversity, the *relay advisor* module should be able to easily obtain a list of participants that are located in different positions, online at present, and capable to play the role of relays in the overlay network. To achieve this, the *neighbor maintainer* module helps to form a large-scale unstructured overlay network in a scalable and distributed way. For the number and quality sake, we propose two techniques, namely the IP-layer routing inference and knowledge sharing, to efficiently pre-select advisable relay candidates without performing active measurement. Section 3 will introduce and evaluate relevant algorithms in details.

2.3. Data plane

In order to make the best use of the application-layer paths provided by the *relay selector* module, CORS has to deal with the following issues in the data plane. First, the multimedia data have to arrive at the destination before its due time to be played out. Secondly, the data dispatched onto each application-layer path should neither exceed its available bandwidth nor overload the relay. Last, proper mechanisms are needed to deal with the unpredictable interrupt of some application-layer paths whose relays are not played by infrastructural nodes and may go offline abnormally.

For the clarity of essential ideas, we assume that the application software of interactive multimedia communications passes the multimedia data fragment (for example a video frame or the payload of a RTP packet) through *CORS API* with explicit requirements on the data's significance and delay. We also assume that the application software uses adaptive multimedia coding schemes, which can adapt its data generation rate to the variation of under-layer available bandwidth.

The E2E performance of each application-layer path in use is characterized by three attributes: E2E delay, transmission reliability, and available bandwidth. By transmission reliability, we mean if the relay is not played by an infrastructural node, then besides the corresponding path's packet loss rate L, the relay's online probability denoted by Q should also be considered. Specifically, an application-layer path's reliability is defined to be $R = (1 - L)Q$. We assume that an end-host's online duration follows an exponential distribution and accordingly compute the end-host's online probability by $Q = e^{-\frac{T_n - T_s}{\tau}}$, where T_n is the present time, T_s is the relay's latest boot time, and τ is the average of the relay's continuous online duration in history.

Initially, an application-layer path's attributes are measured by the *relay selector* module, and then they are continuously updated by the sender's *multi-path manager* module. In particular, obtaining the initial information to calculate application-layer path's transmission reliability and available bandwidth also involves a negotiation with the relay's *packet forwarder* module. After the negotiation, the application-layer path's available bandwidth is assigned to be the minimum of the measured E2E available bandwidth and that the relay agrees to offer.

Invoked by the upper-layer application software to transmit a data segment, the sender's *CORS API* first decides according to the data segment's significance whether it is necessary to process the data segment with proper channel coding schemes such as forward error coding (FEC), which can increase the transmission reliability by adding redundancy.

Then, the sender's *multi-path manager* module properly dispatches the ready-to-send packets to different application-layer paths. Specifically, the *multi-path manager* module searches the list of application-layer paths from its beginning for the first path p that satisfies $T_p \leqslant T_n$ and $S \leqslant B_p t$, where T_p stands for the path's latest idle time, T_n for the present time, S for the packet size, B_p for the path's available bandwidth, and t for the packet's maximum expected transmission time, which can be calculated from the data's delay requirement, the path's E2E delay and the systematic delay caused by the other parts (mainly including the sampling duration on the sender side and the detained duration in jitter buffer on the receiver side).

On sending the packet, path p with its latest idle time updated to be $T_p = T_n + \frac{S}{B_p}$ is moved to the end of the path list. If the transmission reliability R_p of path p is smaller than the specific threshold H corresponding to the packet's significance, the *multi-path manager* module will repeat the above procedure to transmit the packet through multiple application-layer paths simultaneously, until these paths in parallel can ensure an eligible overall transmission reliability $R_\Omega = 1 - \prod_{i \in \Omega}(1 - R_i) \geqslant H$, where Ω stands for the set of the used paths. If unable to find such a set of application-layer paths, which most likely happens in the starting phase of a session, the *multi-path manager* module will also transmit the packet through the default IP-layer path.

As CORS is designed to use only one-hop application-layer paths, the sender's *multi-path manager* module can easily embed the routing information into a custom-defined packet header on the application layer. In order to make the header as short as possible, we can even leverage the IP and UDP header and reuse the same field in CORS header to carry only the necessary IP address and port. Specifically, the sender's *multi-path manager* module embeds the receiver's IP address and port into CORS header and transmits the packet to the relay using UDP protocol; on receiving the packet, the relay's *packet forwarder* module extracts the receiver's IP address and port from CORS header and fills CORS header with the sender's IP address and port, which can be obtained from the source IP address and source port fields in the packet's IP and UDP header, and then forwards the packet to the receiver; finally, the receiver's *multi-path manager* module receives the UDP packet from the relay, recognizes the original sender's IP address and port from the packet's CORS header, puts the payload data in the corresponding session's jitter buffer.

Whenever enough packets that are necessary to reconstructing a data segment have arrived, the receiver's *CORS API* will hand in the reconstructed data segment to the upper-layer application software.

2.4. Deployment

For the flexibility of deployment, we encapsulate the modules of CORS into two separate components (*CORS Proxy* and *Client*

Please cite this article in press as: L. Tang et al., CORS: A cooperative overlay routing service to enhance interactive multimedia communications, J. Vis. Commun. (2009), doi:10.1016/j.jvcir.2009.06.005

ler than some threshold, a number of new neighbor candidates are requested and fetched from currently available neighbors. Such an algorithm is scalable and suitable for overlay networks in which most participants have relatively high churn rates. Given the size of a probe packet is no larger than 50 bytes, the *neighbor maintainer* component can maintain a neighbor list having 600 neighbors and probe every neighbor every 1 min by consuming only 4 Kbps bandwidth.

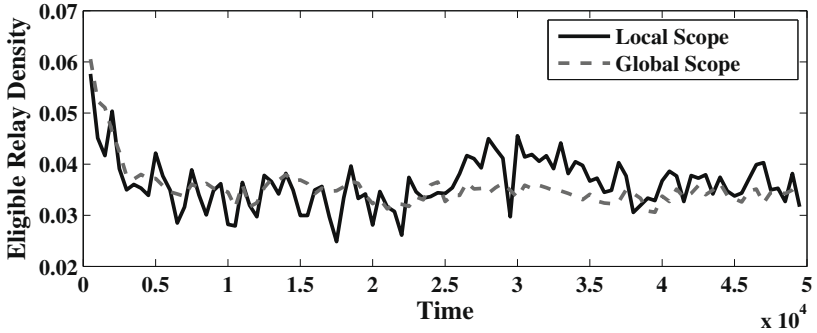
As the neighbor list is the original source of relay candidates to compose application-layer paths, the path diversity is highly dependent on the composition of the neighbor list. In particular, the ratio of eligible relays in the neighbor list is expected to be close to or even larger than that in the whole overlay network. Next, we verify whether or not the above simple neighbor maintenance algorithm is able to accomplish this purpose.

The *eligible relay density* is used to refer to the ratio of eligible relays in a specific scope. Given a particular session, the eligible relay densities in two scopes are of interest. The first scope is the whole overlay network referred to as the *global scope*, and the other is the sender's neighbor list referred to as the *local scope*. In order to check whether or not the average eligible relay density in the local scope can keep close to or larger than that in the global scope, a discrete event simulator is developed.

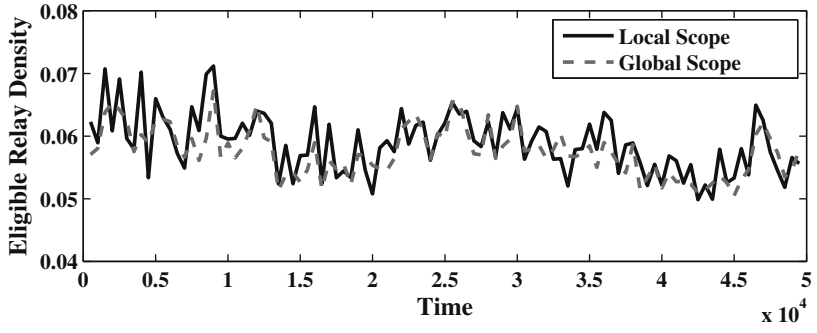
Specifically, we extend BRITE [13] to generate a two-layer topology of the IP-layer networks. The first layer is generated based on the classic BA model [14], where every node stands for an AS in the Internet. The hop count is used as the routing metric to calculate the shortest inter-domain paths. On the second layer, each AS is expanded to an independently generated sub-graph (also based on the BA model) where every node stands for a router in the Internet. In each sub-graph, several nodes are randomly selected to act border routers that connect to the neighboring ASes' border routers. The intra-domain routing metric is the link weight that denotes delay in practice. To obtain the whole IP path between a pair of routers, we first generate the AS path, and then convert it hop-by-hop into the router-level path based on the 'hot-potato' principle, i.e., taking the nearest border router that can get into the next-hop AS as the exit of the current AS. The results presented in this section are based on a typical topology that consists of 1233 routers, which are scattered in 20 ASes.

To generate the overlay network, we randomly scatter a number of end-hosts, each of which attaches to a router in the above topology. The link weight between an end-host and the router it attaches to is randomly assigned to be a small value to analog the effect of different access networks. Two churn models are examined to characterize the participant's join and leave. The first is a widely used synthetic model that considers each participant's online and offline time as two independent random variables following the exponential distribution. The second churn model is generated from the practical trace data of thousands of desktop computers in Microsoft Corporation [15]. An end-host is considered as an eligible relay for a session if simultaneously satisfying two conditions: the end-host is online; the end-host can compose a one-hop application-layer path whose delay is no larger than that of the session's IP path.

The probe interval is set to be 10 time units, the average online and offline time in the synthetic churn model to be both 3600 time units, the statistics interval to be 500 time units, and the total simulation duration to be 50,000 time units. Fig. 3 illustrates when there are 1000 end-hosts in the overlay network, how the local and global eligible relay densities evolve since the simulation starts. The local eligible relay density is calculated as the average of 10 randomly selected sessions. As can be seen, when the simulation gets into the stable state, the neighbor maintenance



(a) Synthetic Churn Model



(b) Practical Churn Model

Fig. 3. The density of eligible relays in the neighbor list keeps close to or larger than that in the whole overlay network, which ensures CORS to achieve path diversity as expected.

Please cite this article in press as: L. Tang et al., CORS: A cooperative overlay routing service to enhance interactive multimedia communications, J. Vis. Commun. (2009), doi:10.1016/j.jvcir.2009.06.005

algorithm can effectively keep the local eligible relay density close to or larger than the global one. This forms the basis for CORS to achieve path diversity in its pre-selection of relay candidates.

Fig. 4 shows the ratios of the local eligible relay density to the global one under different overlay network scales. Clearly as it is, all the ratios keep constantly close to 1.0 as the number of end-hosts increases. It means that the neighbor maintenance algorithm is able to keep the composition of participant's neighbor list homogeneous relative to that of the whole overlay network. Also note that CORS achieves this by costing each participant nearly constant resources, because despite the expansion of the overlay network, the size of each participant's neighbor list is unchanged and each participant continues to probe the same number of neighbors. To sum up, the above simulation results indicate that the neighbor maintenance algorithm of CORS is scalable and capable of working in a large-scale overlay network where the participants have reasonably high churn rates.

3.2. Pre-selection of relay candidates

While the neighbor list provides a homogeneous subset of the whole overlay network, the density of eligible relays in the neighbor list is too low for CORS to directly select relays through active measurement. To deal with this problem, we propose a knowledge-sharing technique to pre-select advisable relay candidates.

The key idea is to share the knowledge among end-hosts that are located closely to each other in the Internet. As the current Internet forwards packets according to the packet's destination IP prefix, the distance between two end-hosts can be measured by the length of their common IP prefix. Intuitively, if the senders of several sessions are located closely (having considerably long common IP prefix) and so are the receivers, it is likely that these sessions will share many common eligible relays.

In order to share the useful knowledge of advisable relay candidates among relevant sessions, the knowledge has to be effectively represented, stored and accessed. To this end, we use a five tuple (*sender, receiver, relay, delay, timestamp*) to represent a knowledge item, where the former three fields, respectively, stand for the sender's, receiver's, and relay's IP prefix, the fourth field is the corresponding application-layer path's E2E delay, and the last field indicates expiry time of this knowledge item. Every knowledge item is indexed by its first two fields and can be stored and accessed by one of the following schemes:

Cache of nodes: In this case, the knowledge from every sender is locally cached in the *CORS Proxy* component's *relay advisor* module. As shown by the pseudo-code in Fig. 5, when other senders query relays, the *relay advisor* module looks through its local cache for IP prefixes of eligible relays that have been verified previously; then, it replies to the query with all its neighbors having such IP prefixes. If there is not enough in the local neighbor list, the *relay advisor* module will delegate the query to other neighbors.

Database: In this case, all knowledge items are stored by and accessed through a centralized database, which provides the *relay advisor* module with a group of SQL-like remote procedure calls.

DHT networks: Both above schemes have disadvantages. The first scheme makes knowledge shared among a relatively small area, while the second one suffers from a single-point failure and bottleneck. Using a fully distributed way to share knowledge in the global scope, DHT networks can alleviate both disadvantages. The negative side using DHT is the increase of complexity and maintenance cost, especially when the nodes have high churn rates. How to leverage DHT's control messages to facilitate the maintenance of CORS overlay network is an important direction for future work.

To evaluate whether and to what extent the knowledge-sharing technique is able to improve the quality of recommended relay candidates, we use the same simulation framework as in the last subsection, but randomly scatter 5000 end-hosts into the topology. End-hosts attached to the same router are considered to have the same IP prefix. With the knowledge-sharing scheme, when a sender requires k relay candidates, it selects from its own neighbor list at most $k-1$ relay candidates based on the shared knowledge, and then complement the rest with the latest probed neighbors. The second part of relay candidates is to ensure the shared knowledge can be constantly refreshed and improved. Eventually, after verifying all the k relay candidates, the sender contributes the best relay's information to improve the shared knowledge. As a comparison reference, we introduced another straightforward pre-selection method named *latest-k*, which simply recommends the latest k probed neighbors as relay candidates. The simulation sets k to be 10 and limits the maximum number of shared relay prefixes per pair of sender and receiver prefixes to be 5.

The first evaluation metric is the ratio of eligible relays out of the relay candidates recommended by different methods. Table 1 shows the central 95% percentile of the sampled eligible relay ratios through the simulation process. It indicates that compared to the *latest-k* method, the knowledge-sharing technique is effective to statistically improve eligible relay density of the recommended relay candidate set. This means that using the knowledge-sharing technique, the *relay advisor* module is able to more precisely recommend eligible relays and thus can save the sender's *relay selector* module from wasting a lot measurement resources.

Another metric referred to as the *relative delay* is introduced to evaluate the quality of the best relay candidate recommended by each method. Given a specific observed sample, i.e., a specific session at a specific simulation time, an application-layer path's relative delay is defined as the ratio of the path's delay divided by the corresponding baseline delay, which is the delay of the shortest application-layer path recommended by the *latest-k* method. Fig. 6 illustrates the cumulative distribution function (CDF) plots of the relative delay, respectively, of the best relay existing in the sender's neighbor list and the best relay recommended by the

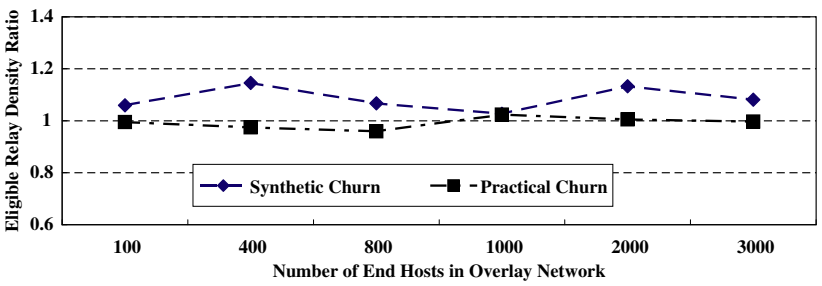
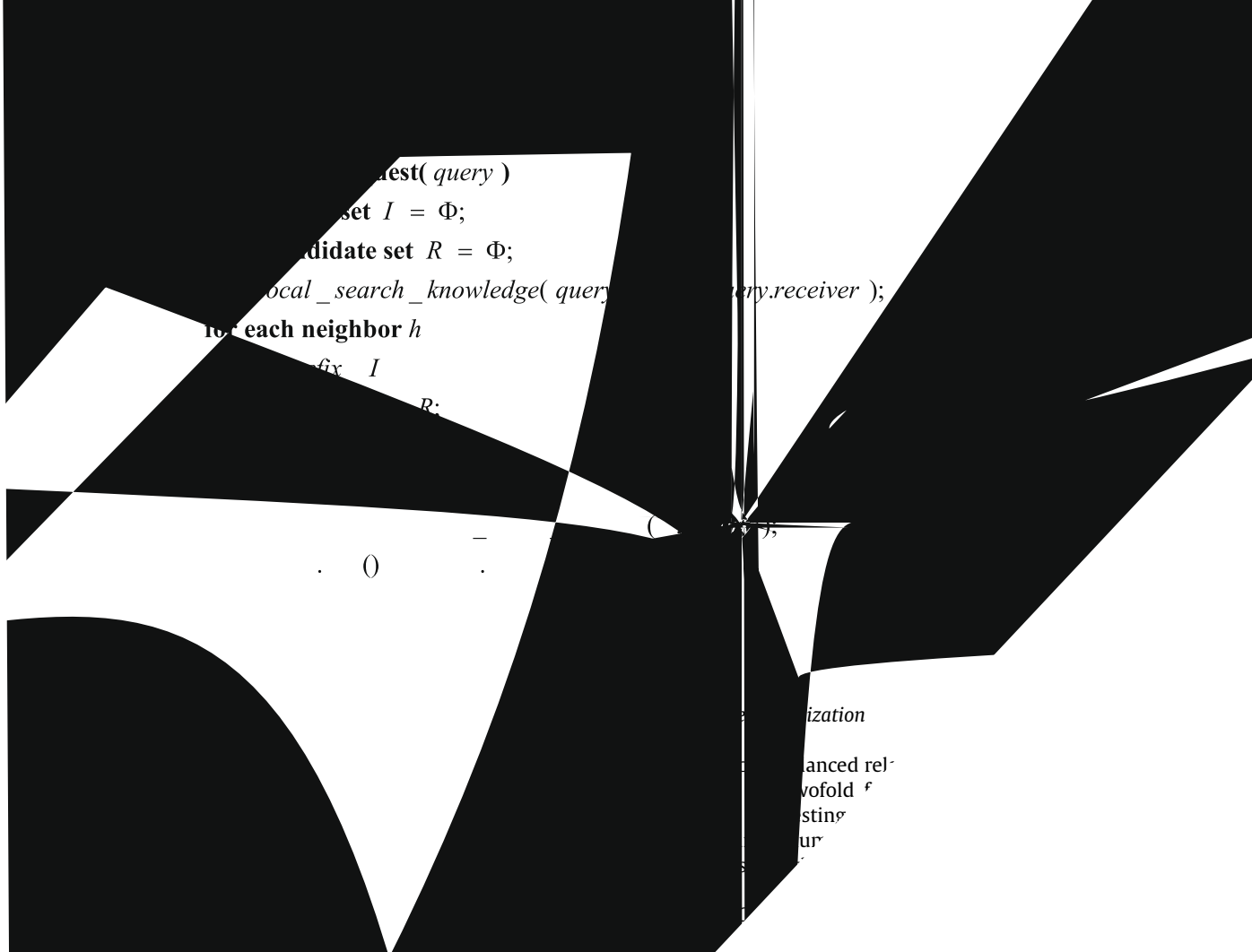


Fig. 4. The average ratio of the local eligible relay density in the neighbor list to the global eligible relay density in the overlay network generally keeps constant as the scale of overlay network increases.

Please cite this article in press as: L. Tang et al., CORS: A cooperative overlay routing service to enhance interactive multimedia communications, J. Vis. Commun. (2009), doi:10.1016/j.jvcir.2009.06.005



prohibitive to globally collect every relay candidate's real-time workload.

To design a distributed algorithm, we propose the redundantly lightest load first' (RLLF) heuristic, which makes the *relay selector* component randomly select more than enough (controlled by the *redundancy factor*) relay candidates and then use them in ascending order of their loads. This intuition is inspired by the 'the power of two choices' paradigm [16]; while simple at the first glance, it proves to be surprisingly effective as shown in the following simulation.

We compare RLLF with two other algorithms RRS and RPD. RRS is the most straightforward algorithm that randomly selects relays out of all available relay candidates until exceeding the bandwidth demand. Like RLLF, RPD also randomly selects more than enough relay candidates, but RPD uses all these selected relays by dispensing the demanded bandwidth in proportional to each relay's currently available bandwidth.

In the simulation, it is assumed there are totally 200 pairs of source and destination end-hosts, each of which generates sessions independently. The session's average duration and interval are, respectively, 30 and 400 time units and both follow the exponential distribution. There are totally 10 available relays, out of which every session can find 5 eligible relay candidates. Every session demands 56 Kbps bandwidth, while every relay can at most dedicate 128 Kbps bandwidth. Every simulation scenario is repeated five times and the average result is used to alleviate accidental biases.

Fig. 7 shows the session failure rate, i.e., the proportion of the failed sessions (due to inadequate bandwidth of the eligible relays) to all generated sessions, under different scenarios. As can be seen, compared to RRS, both RLLF and RPD can effectively reduce the session failure rate thanks to their redundant relay selection. Specifically, with a quite large redundancy factor, RLLF and RPD, respectively, reduce the session failure rate by around 20% and 40%. It also shows that while increasing the redundancy factor can keep helping RLLF and RPD bring down the session failure rate, the redundancy factor's marginal utility actually declines. In practice, using a small redundancy factor, such as 2, is already enough to achieve most improvement of the session failure rate.

Although RPD performs better in terms of reducing the session failure rate, it requires every session to utilize a much larger number of relays, as shown in Fig. 8. This means that RPD will severely increase both the negotiation cost and coordination complexity because of simultaneously using much more application-layer paths. In contrast, by preferring lightest load relays, RLLF can make a session on average require even less relays than if using RRS. Also note that the implementation of RLLF is as simple as that of RRS, we believe RLLF is more suitable than RPD for practical use.

In general, the above results indicate that RLLF with the redundancy factor of 2 is an efficient and practical algorithm for relay utilization and bandwidth allocation, i.e., first randomly selecting a set of relay candidates whose overall usable bandwidth is twice larger than demanded, and then picking the lightest loaded relays to use until satisfying the bandwidth demand.

4. Experiments and evaluations

To evaluate the feasibility and effectiveness of CORS in practice, we implemented a prototype consisting of around 9000 lines C++ source code and deployed it on PlanetLab. This section reports the practical experiment results.

4.1. Experiment setup

In order to avoid subjective interference, we use a 30-s-long foreman video clip of the standard CIF format and 30 frames per second as the content of interactive multimedia communications and use two objective metrics, the data segment loss rate and peak signal-to-noise ratio (PSNR), to compare the transmission effects of the Internet's ordinary transmission service and that of CORS. Because CORS may transmit the same data redundantly through several different application-layer paths, we use the term *data segment loss rate* to distinguish it from the network-layer *packet loss rate*. In regard to the particular multimedia traffic, the video clip is encoded with the H.264/AVC JM 12.4 codec in six different bit rates, namely 128, 256, 384, 768, 1024, and 2048 Kbps, and the group of pictures (GOP) structure is set as 'IBBPBBPBBPBBPBBPBBI'. The generated data segments are encapsulated into RTP packets that are mostly smaller than 1200 bytes.

Considering that interactive multimedia communications have stringent requirements on E2E delay, we assume a video data segment has to be played on the receiver side within 400 ms since the data segment was generated on the sender side. Specifically, every packet in the experiment carries a time stamp, and the packets that fail to arrive within 300 ms are considered to be over-delayed. We do not differentiate the delay variation of the packets that are successfully received within 300 ms, because most application software of multimedia communications implements a jitter buffer to conceal the harmfulness of packet's delay variation. Moreover, we conservatively assume that the overall delay caused by the other parts except the transmission across the Internet is no larger than 100 ms. The clock drift between the sender and receiver is calculated from 10 training packets, whose transmission delay from the sender to the receiver are calculated as half of the minimum of 10 round-trip time (RTT) measurements.

In each run of the experiment, a pair of sender and receiver are randomly selected out of around 300 PlanetLab nodes, and the other PlanetLab nodes are used to play the role of relay candidates. First, the sender sends the encoded video content, respectively, using the ordinary UDP protocol through the direct IP path and using CORS. Then, on the receiver's side, two video clips are separately reconstructed from the usable packets, respectively,

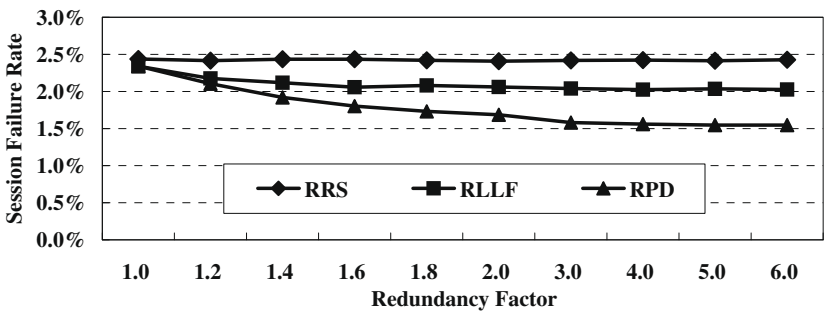


Fig. 7. Session failure rate caused by different relay utilization and bandwidth allocation algorithms.

Please cite this article in press as: L. Tang et al., CORS: A cooperative overlay routing service to enhance interactive multimedia communications, J. Vis. Commun. (2009), doi:10.1016/j.jvcir.2009.06.005

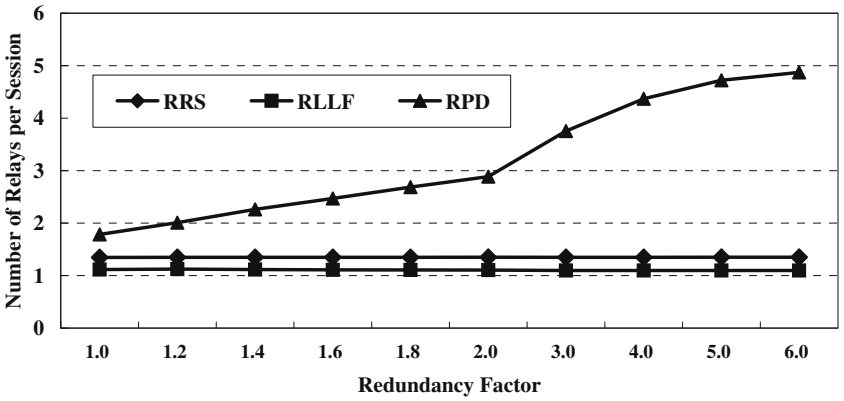


Fig. 8. Average number of relays utilized per session with different relay utilization and bandwidth allocation algorithms.

received by the two different transmission schemes. Finally, the data segment loss rate and PSNR are calculated offline.

4.2. Data segment loss rate

Table 2 shows the statistical comparison between transmitting the video using CORS and using ordinary IP transmission service in term of the resulted data segment loss rate. As can be seen, in a large part of the experiments, there is no data segment lost or over-delayed at all, implying that the sender and receiver are well connected on the IP layer and the IP path's usable bandwidth has already been able to support the video communications.

In most (around 90%) of the other cases, CORS is able to achieve lower data segment loss rate than the ordinary IP transmission service, no matter whether considering the effect of over-delayed packets.

However, there are also a few cases in which CORS has lead to higher data segment loss rate than the ordinary IP transmission service. Most likely, in these experiments the E2E performance bottleneck between the sender and receiver is located either in the sender's or in the receiver's access networks. As all application-layer paths utilized by CORS also have to pass through the sender's and receiver's access networks, they cannot detour the performance bottleneck. On the other hand, the overhead of using application-layer paths aggravates the traffic through the bottleneck and accordingly causes an even higher loss rate of data segments. To solve this problem, CORS can be improved in future by intelligently monitoring the effect of using application-layer paths or developing measurement techniques to locate the performance bottleneck.

Fig. 9 illustrates the average data segment loss rates without considering the *No-Damage* cases that do not suffer any lost or over-delayed packets at all. Under all the investigated six bit rates of the encoded video clip, CORS can always remarkably reduce the percents of lost and over-delayed data segments compared with the ordinary IP transmission service. The results indicate that CORS is effective to increase the transmission reliability by utilizing multiple application-layer paths.

Table 2
Statistics on the comparisons between using CORS and using the ordinary IP transmission service in terms of the data segment loss rate with/without the over-delayed packets, where *No-Damage* stands for the cases that neither transmission scheme causes any data segments lost or over-delayed and *Others* for the cases that using the ordinary IP transmission service is better than or equivalent to using CORS.

Bit rate (Kbps)	# Experiments	% No-Damage	% CORS-Better	% Others
128	498	80.1	18.3/19.5	1.6/0.4
256	468	77.8	20.3/20.9	1.9/1.3
384	453	72.6	25.6/25.8	1.5/2.2
768	493	72.4	25.4/25.8	2.8/1.9
1024	427	69.3	27.9/28.8	1.9/2.0
2048	399	70.7	27.3/27.1	2.0/2.2

4.3. Peak signal-to-noise ratio

In calculating the data segment loss rate, every data segment is treated equivalently. In fact, however, as has been indicated, different data segments in interactive multimedia communications may have quite different significance and thus different impacts on the quality and user experience. Taking the H.264 coding standard for example, data segments carrying I-frames are usually much more important than those carrying P-frames or B-frames.

To gain a deeper insight on whether and to what extent CORS can improve the quality of multimedia communications, we also compare the PSNRs, respectively, achieved by CORS and by the ordinary IP transmission service, since PSNR is an objective video evaluation metric that has been widely used in literature. Given a pair of the original video for reference and the reconstructed video to evaluate, the corresponding PSNR is defined as $PSNR = 10 \lg \frac{NL^2}{\sum_{i=1}^{N}(x_i - y_i)^2}$, where N stands for the number of pixels in the video, L for the maximum value of a pixel, x_i and y_i, respectively, for values of the ith pixel of the original and evaluated video. The larger is PSNR, the less degradation happens to the video quality. Usually, people can feel about difference if the change of PSNR is larger than 0.5 db [17].

The H.264/AVC JM 12.4 codec is used to reconstruct the video on the receiver side from the usable data segments that are successfully received. The codec's error concealment mechanism is set as the *Frame Copy* mode, which would replay the last frame if the current frame could not be reconstructed. However, there are still a small part of sessions for which the JM codec cannot reconstruct the video according to the received data segments, because these sessions have lost too many or some very significant data segments.

Table 3 shows the statistical comparison results between transmitting the video, respectively, with CORS and with the ordinary IP transmission service in term of PSNR. Similar to the metric of data segment loss rate, there is also a majority of cases that the E2E performance between the sender and receiver is good enough to meet the transmission requirements and therefore the transmission procedure merely causes imperceptible harmfulness to the video's quality, especially when the video is encoded in relatively low bit rates. In most of the other cases, CORS outperforms the ordinary IP transmission service.

Excluding the *Imperceptible* cases, Fig. 10 illustrates the average PSNR, respectively, achieved by CORS and by the ordinary IP trans-

Please cite this article in press as: L. Tang et al., CORS: A cooperative overlay routing service to enhance interactive multimedia communications, J. Vis. Commun. (2009), doi:10.1016/j.jvcir.2009.06.005

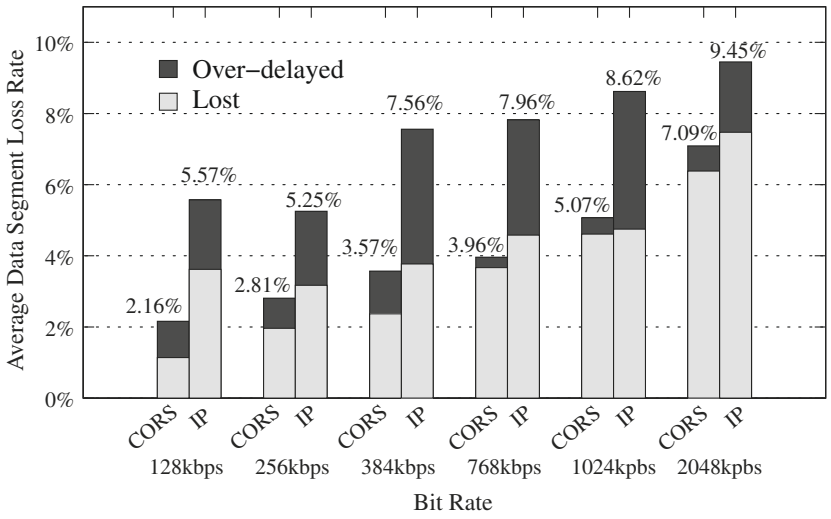


Fig. 9. Average data segment loss rates when transmitting, respectively, with CORS and the ordinary IP service, where the *No-Sense* cases are excluded out.

Table 3
Statistics on the comparison between using CORS and using the ordinary IP transmission service in terms of PSNR, where *Imperceptible* stands for the cases that no matter using CORS or the ordinary IP service, the difference between the resulted PSNR and the maximum PSNR that can be achieved by the corresponding coding scheme and bit rate is too small (less than 0.5 db) to be perceptible, *CORS-Better* for the cases that the PSNR of using CORS is notably better, i.e., more than 0.5 db larger, than that of using the ordinary IP service.

Bit rate (Kbps)	# Decodable	% Imperceptible	% CORS-Better	% Others
128	424	99.1	0.9/0.9	0.0/0.0
256	417	95.2	3.6/3.6	1.2/1.2
384	419	90.2	9.1/9.1	0.7/0.7
768	462	89.0	9.9/10.1	1.1/0.9
1024	405	87.4	10.9/11.4	1.7/1.2
2048	377	86.5	13.0/13.0	0.5/0.5

mission service. As can be seen, CORS can effectively improve the average PSNR compared to the ordinary IP transmission service. Besides reducing the data segment loss rate and emphasizing the protection of significant data segments, another reason enabling CORS to achieve higher PSNR is because that CORS successfully alleviates continuous packet loss. For similar reasons, it is noted that the over-delayed data segments usually have smaller impact on the video's PSNR than the lost data segments. The distribution of over-delayed data segments is relatively uniform, while the distribution of lost data segments exhibits a burst and clustered pattern. Therefore, the continuously lost data segments have higher probability to cause undecodable frames than the dispersively over-delayed data segments.

While not included in above statistics, we also observe that the IP paths between several pairs of sender and receiver were completely interrupted, but CORS enabled them to communicate through the application-layer paths. A typical example is that CORS has used the relays 'lsirextpc02.epfl.ch' and 'peeramide.irisa.fr' to successfully bypass the IP-layer interrupt between the sender 'pli2-pa-1.hpl.hp.com' and the receiver 'planetlab-02.bu.edu'.

5. Related work

Generally speaking, the techniques for improving the quality of multimedia communications can be classified into three catego-

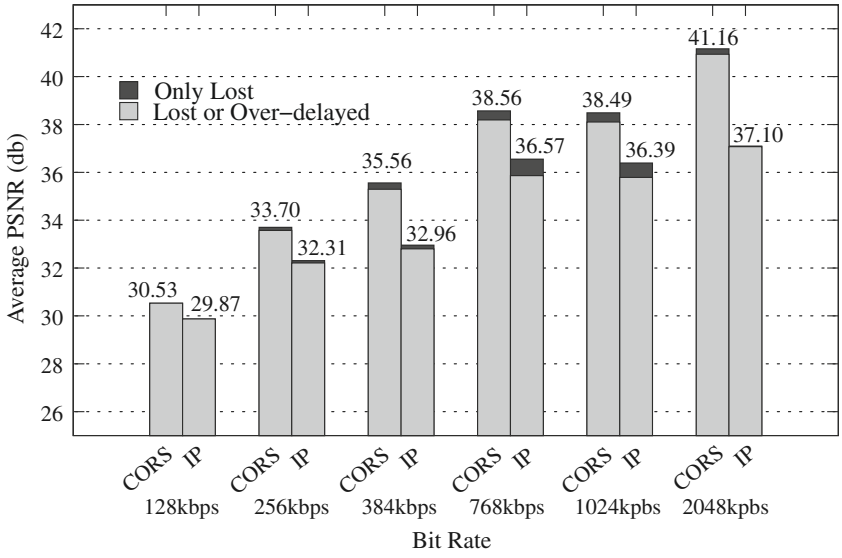


Fig. 10. Average PSNR of the reconstructed video on the receiver side, respectively, using CORS and the ordinary IP transmission service.

Please cite this article in press as: L. Tang et al., CORS: A cooperative overlay routing service to enhance interactive multimedia communications, J. Vis. Commun. (2009), doi:10.1016/j.jvcir.2009.06.005

ries: IP-layer QoS enhancement such as Intserv [18] and Diffserv [19], advanced coding schemes [20,21] such as FEC [22,23] and MDC [11], and overlay routing techniques. In this paper, as CORS focuses on designing the architecture and pivotal algorithms to find a diversity of eligible application-layer paths, it is most related to the other overlay routing systems.

The emergence and development of routing overlays were mostly inspired by the research results of the Internet's E2E performance and routing behaviors. Much work has revealed that the current IP-layer routing service was far from ideal to generate optimal E2E paths [24–26], and in majority cases, there was an alternate path notably superior to the direct IP path [2–4].

RON [10] was one of the earliest systems using overlay routing techniques to improve the Internet's E2E availability and performance. RON constructed a full-meshed overlay network in which every node actively detected its virtual-link quality to every other node; a modified link-state routing protocol was used to disseminate the topology and calculate the route on the application layer. Accordingly, every node in RON had to spend a large amount of bandwidth in actively monitoring the virtual-link's quality. It was acknowledged that RON was not scalable enough to support overlay networks including more than a hundred nodes.

To improve scalability, a two-level hierarchy was introduced in Spines project [27]. Similar to RON, Spines also required the router nodes to monitor the link's quality and run a link-state-like routing protocol on the application layer. However, when a client application intended to leverage Spines, the client did not have to host a router node itself, instead it could connect to a router node that acted as a proxy for the client to send and receive packets through Spines networks. The specific use of Spines for improving the quality of VoIP streams was proposed and studied in [28,29].

CORS differs from RON and Spines on several aspects. First, both RON and Spines made use of overlay routing techniques to improve the E2E path reliability and performance for all types of Internet applications, but CORS aims to build a transport-layer service particularly suitable for interactive multimedia communications. Specifically, CORS supports to differentiate the significance of multimedia data segments and accordingly transmit them through different paths, but RON and Spines did not. Moreover, RON and Spines organized overlay networks essentially the same as the Internet's IP-layer intra-domain routing protocols, all based on a proactive link-state single-path routing, which inherently was not scalable to support large-scale overlay networks. In contrast, CORS constructs one-hop application-layer paths reactively; it not only leverages the large scale of overlay networks to achieve path diversity, but also can coordinate multiple paths to overcome the limit of IP-layer's single path routing service.

CORS borrows the idea of using one-hop application-layer paths from SOSR [8], and also stems from much previous research revealing that most performance gains of overlay routing could be achieved by utilizing a single relay node [3,4]. However, unlike SOSR that focused on bypassing the Internet's path failure and improving E2E availability, CORS aims to improve the E2E performance and quality of interactive multimedia communications. Therefore, while SOSR could use randomly chosen relays to achieve its goal quite easily, CORS has to design elaborate techniques to make the selected relays satisfy the requirements on path diversity and E2E performance.

There have also been proposals using the Internet's topological heuristics for designing overlay routing protocols. ASAP presented an AS-aware overlay routing protocol to improve Skype's inefficient relay selection and to achieve high-quality VoIP service [30]. ASAP required infrastructural nodes to build up-to-date AS graph by collecting publicly available BGP snapshots and update messages. The AS graph was used to cluster end-hosts and conduct surrogate nodes to maintain a set of close clusters. To find eligible application-layer paths for a session, ASAP selected one-hop and two-hop relays by intersecting the source's close cluster sets and the destination's ones. Different from ASAP, CORS develops the knowledge-sharing technique to enable end-hosts that are located closely to each other to cooperate and share their experiences and knowledge about eligible relays.

6. Conclusions and future work

As more and more applications of interactive multimedia communications are emerging and developing rapidly on the Internet, this paper identifies the inherent limits of the current Internet's IP-layer routing service to support high-quality interactive multimedia communications.

To deal with the challenges, we propose a cooperative overlay routing service named CORS. Essentially, CORS aims to provide a flexible multi-path transmission service to interactive multimedia communications by means of composing and coordinating a number of eligible application-layer paths. We particularly focus on elaborating the architecture and design rationales of CORS in this paper. Unlike previous overlay routing systems using the same proactive routing style as the IP-layer intra-domain routing protocols, CORS composes application-layer paths in a reactive manner, which can significantly save the cost of continuously measuring the quality of virtual-links in overlay networks and therefore increases scalability. We enable CORS to achieve path diversity by maintaining a large-scale overlay network in a distributed and scalable way, and exploit a novel knowledge-sharing technique to efficiently recommend agreeable relay candidates. Finally, a lightweight relay selection and bandwidth allocation algorithm is proposed to help CORS balance the load and make good use of available relays. Simulation results show that the proposed algorithms are effective and scalable.

CORS has been developed as a prototype on PlanetLab. The experiment results verify the feasibility and effectiveness of CORS on improving the transmission reliability and quality of interactive multimedia communications. In future work, our ultimate goal is to implement CORS as an easy-to-use library with a set of well-defined socket-like APIs.

Acknowledgements

The authors thank the colleagues in Network Security Lab, RIIT, Tsinghua University for their cooperation on developing CORS, and are very grateful to Dr. Yong Xia at NEC and anonymous reviewers for their insightful comments and suggestions on improving this paper. This work is sponsored by NEC Laboratories, China.

References

[1] One way transmission time, ITU-T Recommendation G.114, March 2005.
[2] S. Savage, A. Collins, E. Hoffman, J. Snell, T. Anderson, The end-to-end effects of internet path selection, ACM SIGCOMM Computer Communication Review 29 (4) (1999) 289–299.
[3] H. Rahul, M. Kasbekar, R. Sitaraman, A. Berger, Towards realizing the performance and availability benefits of a global overlay network, in: Proceedings of Passive and Active Measurements (PAM) Conference, 2006.
[4] H. Zhang, L. Tang, J. Li, Impact of overlay routing on end-to-end delay, in: Proceedings of 15th International Conference on Computer Communications and Networks, 2006.
[5] V. Subramanian, S. Kalyanaraman, K.K.Ramakrishnan, Hybrid packet FEC and retransmission-based erasure recovery mechanisms (HARQ) for lossy networks: analysis and design, in: Proceedings of Wireless Systems: Advanced Research and Development (WISARD), 2007.
[6] W. Jiang, H. Schulzrinne, Modeling of packet loss and delay and their effect on real-time multimedia service quality, in: Proceedings of 10th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), 2000.
[7] M. Dahlin, B. Chandra, L. Gao, A. Nayate, End-to-end WAN service availability, IEEE/ACM Transactions on Networking 11 (2) (2003) 300–313.

Please cite this article in press as: L. Tang et al., CORS: A cooperative overlay routing service to enhance interactive multimedia communications, J. Vis. Commun. (2009), doi:10.1016/j.jvcir.2009.06.005

[8] K.P. Gummadi, H.V. Madhyastha, S.D. Gribble, H.M. Levy, D. Wetherall, Improving the reliability of internet paths with one-hop source routing, in: Proceedings of 6th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2004.
[9] H. Zhang, J. Zhou, J. Li, M2FEC: an effective FEC based multi-path transmission scheme for interactive multimedia communication, Journal of Visual Communication and Image Representation, this issue, doi:10.1016/j.jvcir.2009.07.001.
[10] D. Andersen, H. Balakrishnan, F. Kaashoek, R. Morris, Resilient overlay networks, ACM SIGOPS Operating Systems Review 35 (5) (2001) 131–145.
[11] V.K. Goyal, Multiple description coding: compression meets the network, IEEE Signal Processing Magazine 18 (5) (2001) 74–94.
[12] Netfilter. Available from: <http://www.netfilter.org/>.
[13] A. Medina, A. Lakhina, I. Matta, J. Byers, BRITE: an approach to universal topology generation, in: Proceedings of 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2001.
[14] A.-L. Barabsi, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.
[15] W.J. Bolosky, J.R. Douceur, D. Ely, M. Theimer, Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs, in: Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), 2000.
[16] M.D. Mitzenmacher, The power of two choices in randomized load balancing, IEEE Transactions on Parallel and Distributed Systems 12 (10) (2001) 1094–1104.
[17] D. Salomon, Coding for Data and Computer Communications, Springer, 2005.
[18] R. Braden, D. Clark, S. Shenker, Integrated Services in the Internet Architecture: an Overview, rfc 1633, June 1994.
[19] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An Architecture for Differentiated Services, rfc 2475, December 1998.
[20] H. Song, C.-C. Kuo, Rate control for low-bit-rate video via variable-encoding framerates, IEEE Transactions on Circuits and Systems for Video Technology 11 (4) (2001) 512–521.
[21] J.C. Wenxian Yang, King Ngi Ngan, An MPEG-4 compatible stereoscopic/multiview video coding scheme, IEEE Transactions on Circuits and Systems for Video Technology 16 (2) (2006) 286–290.
[22] C. Wang, D. Sklar, D. Johnson, Forward error-correction coding, Aerospace Corporation Magazine of Advances in Aerospace Technology 3 (1) http://www.aero.org/publications/crosslink/winter2002/04.html.
[23] D. Wu, Y. Thomas, H.J. Yao, H.J. Chao, Transmission of real-time video over the internet: a big picture, in: Proceedings of IEEE NetWorld+Interop, 2000.
[24] V. Paxson, End-to-end routing behavior in the internet, IEEE/ACM Transactions on Networking 5 (5) (1997) 601–615.
[25] S. Neil, M. Ratul, A. Thomas, Quantifying the causes of path inflation, in: Proceedings of ACM SIGCOMM Conference, Karlsruhe, Germany, 2003.
[26] C. Labovitz, A. Ahuja, A. Bose, F. Jahanian, Delayed internet routing convergence, IEEE/ACM Transactions on Networking 9 (3) (2001) 293–306.
[27] Spines. Available from: <http://www.spines.org/>.
[28] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, A. Terzis, 1-800-OVERLAYS: using overlay networks to improve VoIP quality, in: Proceedings of 10th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), 2005.
[29] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, A. Terzis, An overlay architecture for high-quality VoIP streams, IEEE Transactions on Multimedia 8 (6) (2006) 1250–1262.
[30] S. Ren, L. Guo, X. Zhang, ASAP: an AS-aware peer-relay protocol for high quality VoIP, in: Proceedings of 26th IEEE International Conference on Distributed Computing Systems (ICDCS), 2006.

Please cite this article in press as: L. Tang et al., CORS: A cooperative overlay routing service to enhance interactive multimedia communications, J. Vis. Commun. (2009), doi:10.1016/j.jvcir.2009.06.005