Traffic engineering using overlay network

Ryoichi Kawahara¹, Shigeaki Harada², Noriaki Kamiyama¹, Tatsuya Mori¹, Haruhisa Hasegawa¹, Akihiro Nakao³
¹NTT Service Integration Laboratories, NTT Corporation, Tokyo 180-8585, Japan
²NTT West Corporation, Japan
³Graduate School of Interdisciplinary Information Studies, The University of Tokyo, Japan

Abstract—Due to integrated high-speed networks accommodating various types of services and applications, the quality of service (QoS) requirements for those networks have also become diverse. The network resources are shared by the individual service traffic in the integrated network. Thus, the QoS of all the services may be degraded indiscriminately when the network becomes congested due to a sudden increase in traffic for a particular service if there is no traffic engineering taking into account each service’s QoS requirement. To resolve this problem, we present a method of controlling individual service traffic by using an overlay network, which makes it possible to flexibly add various functionalities. The overlay network provides functionalities to control individual service traffic, such as calculating the optimal route for each service’s QoS requirement. Specifically, we present a method of overlay routing that is based on the Hedge algorithm, an online learning algorithm to guarantee an upper bound in the difference from the optimal performance. We show the effectiveness of our overlay routing through simulation analysis for various network topologies.

I. INTRODUCTION

Network integration based on IP technologies is progressing, and accordingly, various types of services are being aggregated into networks. However, network usage patterns and applications, e.g., peer-to-peer file sharing, customer generated media, video content sharing, and streaming, are becoming more diverse, as is the traffic generated by such applications [1], [2]. Access networks and end host terminals are also becoming more diverse, and their bandwidths and/or processing capacities are increasing. In addition, with the various types of rich content deliveries expected in the future, it is becoming harder to tame traffic patterns in the space domain, e.g., a traffic matrix is less controllable, as well as in the time domain, e.g., temporal cyclic patterns of traffic are now less predictable than ever.

Under such circumstances, because the network resources are shared by the individual service traffic in an integrated network, the QoS of all the services may become degraded indiscriminately when the network becomes congested due to a sudden increase in traffic for a particular service. This may occur if we do not perform any traffic engineering taking into account each service’s QoS requirement. To resolve this problem, we need a traffic engineering method that does take those various requirements into account.

For establishing such a method, we focus on overlay network technology. An overlay network is defined as a logical network that is constructed in order to achieve certain functionalities over a physical network, such as an IP network. Specifically, the overlay network has been attracting attention as a way of adding to any network new functionalities that are hard to implement in a physical network [3], [4]. In this paper, we thus present an architecture that makes use of the dynamic and flexible nature of an overlay network to control individual service traffic. In our architecture, a network provider such as an ISP or a carrier constructs an overlay network with functionality to control individual service overlay traffic. This overlay network provides a platform that makes it possible to dynamically construct overlay topologies and overlay routes for individual services, such as VoIP, streaming, content distribution, and file-sharing, taking into account the services’ QoS requirements. This traffic control is illustrated in Fig. 1, where the physical network consists of nodes (routers) 1 to 6. The delay increases on the link between nodes 1 and 2, while simultaneously the throughput decreases on the link between nodes 3 and 5. In this case, we construct two overlay networks; one is for service X, which is sensitive to the delay metric, and the other is for service Y, which is sensitive to the throughput metric. On each service overlay network, we calculate the optimal route in terms of the corresponding metric.

As shown in Fig. 1, by achieving functionalities for QoS measurement, logical network construction, and route calculation through use of the overlay network, we should be able to control individual service traffic according to the service’s QoS requirements. In addition, if the same content is delivered to many users, we can reduce the traffic generated by the content by caching the content in the network. That is, by adding a caching functionality to the overlay node and caching the content in service Y as shown in Fig. 1, we can deliver the content from the nearby caching overlay node to the request node. This makes it possible to reduce the traffic compared with the case where the content is delivered along a path with many hop counts from the original server to the request node.

In addition to presenting an architecture for controlling
service traffic by using an overlay network, we also present an overlay routing method that makes it possible to avoid performance degradation caused if each service overlay network performs routing control selfishly to improve its own QoS. The proposed method learns better routes by using combinations of past route selection and the resulting traffic/QoS states. This algorithm can provide better routes for individual service traffic with high probability, without assuming traffic patterns for all services in advance. The proposed algorithm is based on the Hedge algorithm [5], an online learning algorithm, which provides theoretical guidelines for turning knobs for targeted performance. In spite of the simplicity of the algorithm, the bound for the difference from the optimal performance is comparable to that of other algorithms. We show the effectiveness of our overlay routing through simulation analysis for various network topologies.

The paper is organized as follows. Section II reviews related work. In Section III, we describe an overlay-network-based traffic control architecture and functionalities needed at individual network equipment in the architecture. In Section IV, we present an overlay routing method. In Section V, we show the effectiveness of our overlay routing through simulation analysis for various types of network topologies. Finally, we conclude in Section VI.

II. RELATED WORK

There has been much work done on overlay-enabled data dissemination. However, most of the work assumes that the overlay network is constructed on top of the Internet and that the overlay and the underlying network do not interact. Instead, the overlay network estimates the QoS using active measurement and tries to improve its own service’s QoS; it does not optimize the entire performance including QoS requirements of individual service overlay networks and the underlying network resource utilization efficiency.

For example, some reports show that we can improve end-to-end QoS by using overlay routing through QoS measurement data on the Internet [6], [7], [8], [9]. A QoS-aware routing overlay network [3] allocates at least one overlay node in each AS domain, measures QoS (available bandwidth) between pairs of overlay nodes, and establishes overlay routes using measured QoS. A resilient overlay network [4] uses overlay routing to improve resiliency and shortens the recovery time from link/node failure compared with the existing border gateway protocol. Ren et al. [10] proposed a method of selecting an overlay relay node for a source-destination node pair so that the delay of the overlay route is less than a threshold. In this method, overlay nodes are clustered using AS information, which makes it possible to avoid exhaustive searches from all available nodes. Similarly, Fei et al. [11] used AS information to construct overlay disjoint paths.

In contrast to these works, the key idea in our architecture design is to enable cooperation between overlay networks and the underlay network to control the traffic of the individual service overlay networks so that we can improve application performance as well as resource utilization of the underlay network. By utilizing the underlay traffic/QoS information, e.g., network topology, each link utilization, and QoS matrix, the overlay network constructs each service overlay topology and then calculates the optimal routes for the service, taking account of the QoS requirements as well as efficient underlying network resource utilization.

To achieve traffic engineering, use of Multiprotocol Label Switching may be a possible way. For example, Refs. [12], [13] establish label switch paths (LSPs) between possible pairs of edge nodes so that each link utilization is balanced. Active measurements along LSPs are used to balance the traffic and improve the QoS in Refs. [14], [15]. However, if we try to optimize individual service traffic in a large-scale network, we need to establish a large amount of LSPs, i.e., $O(n^2 \times m)$, where $n$ is the number of edge nodes and $m$ is the number of services, which is neither cost-effective nor scalable. In contrast, in our architecture, each service overlay can construct its own overlay topology and routes, independently of other service overlays. Another merit of using overlays is that, because the overlay networks are built at the same layer as the applications, it is easy to create a content cache, which will become an inevitable functionality in the future because eliminating redundant traffic due to distributions of the same content can drastically reduce the traffic.

III. OVERLAY-NETWORK-BASED TRAFFIC CONTROL ARCHITECTURE

We propose that a network provider such as an ISP or a carrier constructs an overlay network as a platform with functionality to control individual service traffic. The overlay network consists of overlay nodes prepared by a network provider (nOL nodes), end-host overlay nodes (eOL nodes), and a controller managed by the network provider, as shown in Fig. 2. The nOL node and the controller are prepared by the network provider, while an eOL node is an end-host that receives a service.

The geographically distributed nOL nodes cooperate with the controller to control each service overlay network and the routes. Here, a service corresponds to file-sharing, a streaming service, VoIP service, and so on.

Specifically, the controller manages the traffic/QoS information in an underlay network, e.g., network topology, link utilization, and QoS metrics such as delay, packet loss ratio, and throughput. This kind of information is obtained through
active measurements between overlay nodes, or traffic data measured in the physical network [16]. The information is used in common for all the services. By doing this, we can avoid duplicate traffic measurements between overlay nodes that consist of different service overlay networks.

With the traffic/QoS information in the underlay network, nOL nodes construct each service overlay network topology, such as for services A and B as shown in Fig. 2. They then calculate the optimal routes for the service, taking the QoS requirement into account. The overlay topology determines which nOL nodes are connected logically to the other nOL nodes, where “node X is logically connected to node Y” means that node X knows the IP address of node Y and can communicate with node Y. Establishing a full-mesh overlay topology is not always necessary. For example, if we provide a content delivery service where the content popularity depends on the locality of the users, a localized topology may be enough.

We here describe how packets are relayed along a designated route. First, eOL node a joins a service overlay and accesses the nearest nOL node A, which is a component of the service overlay. Then, nOL node A provides eOL node a with an overlay route to the destined eOL node b. If an overlay route notified to eOL node a is A-C-B, where nOL node C is a relay node and nOL node B is a destination nOL node to which eOL node b belongs, then eOL node a sends packets to relay nOL node C, where these packets have information that the final destination is eOL node b [17].

If the traffic is cacheable, i.e., the same content is delivered to many users, we can reduce the content traffic by caching the content at the nOL nodes. That is, we can deliver the content from a nearby caching nOL node to the request user, which makes it possible to reduce the traffic compared with the case where the content is delivered along a path with many hop counts from the original server.

In the case of applications where eOL nodes can relay and/or cache packets, the eOL nodes cache the received pieces of content and forward them to their neighbors, determined by the nOL nodes. This makes it possible to localize the traffic, i.e., to reduce the traffic exchanged between different areas.

IV. OVERLAY ROUTING ALGORITHM

As explained in Section III, in our architecture, each service consists of its own service overlay network topology and controls the traffic on the topology. Specifically, each service overlay network executes routing control to improve its own QoS, such as latency, packet loss ratio, and throughput. In this case, because there are many service overlay networks sharing the network resources (i.e., bandwidth) of the underlay network, the traffic from multiple service overlays may concentrate on a particular node(s) and/or link(s), which causes performance degradation for the service overlay networks. In fact, some reports such as Refs. [18], [19] indicate the problem of selfish routing. That is, if individual overlay networks determine the overlay routes to improve their own performances, the resulting performances may worsen because the traffic from the individual overlay networks is routed to the same link and causes congestion.

We thus developed a route selection method that makes it possible to avoid performance degradation caused by selfish routing. The proposed method learns better routes by using combinations of past route selection and the resulting traffic/QoS states. This algorithm can provide better routes for individual service traffic with high probability, without assuming traffic patterns for all services in advance. The procedure of our algorithm is as follows.

Let us consider the route from nOL node s to nOL node d. When a flow from an eOL node belonging to nOL node s to that belonging to nOL node d is generated, we first select nOL nodes that are each connected to nOL node s as well as nOL node d as relay node candidates. Here, a flow corresponds to a series of packets composing a download file, for example.

We denote the set of relay node candidates asVs,d and the candidate relay node k∈Vs,d. In addition, if an overlay link is established between nOL nodes s and d, the direct route is also a candidate for route selection.

We denote the metric of each physical link m measured at the ith interval as γ(m), where the measurement interval length is set to a predetermined length of τ, e.g., τ = 1 min. Examples of the metric are link utilization, delay time, and throughput.

We use physical link metric γ(m) of physical link m to determine the link metric of overlay link Ls,i between nOL nodes i and j. As an example, we illustrate the case where γ(m) represents link utilization. Let qs(Ls,j) be the overlay link metric for link Ls,j at the ith interval, and calculate it as

\[ q_s(L_{s,j}) = \max_{m \in S_{s,j}} \gamma(m), \]

where Ss,j is the set of physical links on the route between a pair of nOL nodes s and j.

Using qk(Ls,k) and qk(Lk,d), we determine the route from nOL node s to d. We calculate the route metric rk(qk(Ls,k),qk(Lk,d)) from nOL node s to d via k as

\[ r_k(q_k(L_{s,k}),q_k(L_{k,d})) = \max\{q_k(L_{s,k}),q_k(L_{k,d})\}, \]

which means the highest link utilization on the route. As for the direct route from nOL node s to d, we define the route metric as rd(qs(Ls,s),qs(Ls,d)) = qs(Ls,d).

Note that, if γ(m) is delay for example, qk(Ls,j) is the sum of γ(m) along the path between nOL nodes s and j.

We use a measured performance metric, rk, as expert advice, to choose the route. Specifically, we propose choosing the route by using the Hedge algorithm, which is an online learning algorithm for prediction with expert advice [5]. As we shall show in short, the striking advantage of using the Hedge algorithm lies in the fact that, in spite of its simplicity, the bound for the loss from the best strategy is comparable to that of other algorithms [5].

We assume that T flows are generated between overlay nodes s and d over the period on which we are focusing.
On the basis of the Hedge algorithm, we formulate our route selection scheme as follows:

Step 1) Initially, the first flow randomly chooses route \( k \); i.e., the flow randomly chooses nOL node \( \tilde{V}_{s,d} \) as a relay node, where \( \tilde{V}_{s,d} = V_{s,d} \cup s \) if the direct route is also a candidate, otherwise \( \tilde{V}_{s,d} = V_{s,d} \).

Step 2) When the \( n^{th} \) flow \((n = 2, \cdots, T)\) is generated, the latest measurement metric \( q_t(L_{i,t}) \) is used to calculate the probability of the flow choosing node \( k \in \tilde{V}_{s,d} \) as a relay node, \( p_{s,d,k,n} \):

\[
p_{s,d,k,n} = \frac{p_{s,d,k,n-1} \times \beta^{rt(1_{(q_t(L_{i,t}) > q_t(L_{d,t})})}}}{\sum_{k \in \tilde{V}_{s,d}} p_{s,d,k,n-1} \times \beta^{rt(1_{(q_t(L_{i,t}) > q_t(L_{d,t})})}}}, \tag{3}
\]

where \( \beta \) is a prespecified parameter, which lies in \( 0 < \beta < 1 \). Equation (3) is based on the Hedge algorithm, and route selection probability \( p_{s,d,k,n} \) is iteratively updated using the metric \( r_t \). The smaller the metric \( r_t \), the higher the probability \( p_{s,d,k,n} \). That is, the algorithm chooses the route of a small \( r_t \) with high probability. \( \beta \) determines the degree to which the measured metric \( r_t \) is reflected to choose the route. We explain about an appropriate setting of \( \beta \) later with Eq. (7).

Note that we initially set \( p_{s,d,k,1} = 1/|\tilde{V}_{s,d}| \), where \( |\tilde{V}_{s,d}| \) is the number of candidates belonging to \( \tilde{V}_{s,d} \).

Step 3) The \( n^{th} \) flow chooses the route \( s-k-d \), i.e., from node \( s \) to \( d \) via \( k \), with probability \( p_{s,d,k,n} \). Steps 2 and 3 are repeated every time a flow arrives.

The above Hedge algorithm has the following property. Let \( C_{s,d,n} \) be the cost (= route metric) that the \( n^{th} \) flow from nodes \( s \) to \( d \) observes. The probability of the \( n^{th} \) flow choosing the route \( s-k-d \) is \( p_{s,k,d,n} \), and the cost of the corresponding \( n^{th} \) flow choosing the route \( s-k-d \) is \( r_t(q_t(L_{i,k}), q_t(L_{d,k})) \). That is,

\[
Pr(C_{s,d,n} = r_t(q_t(L_{i,k}), q_t(L_{d,k}))) = p_{s,d,k,n}. \tag{4}
\]

The expectation of \( C_{s,d,n} \) for \( T \) flows, \( E(\sum_{n=1}^{T} C_{s,d,n}) = \sum_{n=1}^{T} \sum_{k} p_{s,d,k,n} r_t(q_t(L_{i,k}), q_t(L_{d,k})) \), is bounded as

\[
E(\frac{\sum_{n=1}^{T} C_{s,d,n}}{T}) - \min_k \sum_{n=1}^{T} r_t(q_t(L_{i,k}), q_t(L_{d,k}))/T \leq \frac{1-\beta}{2\beta} \cdot \frac{M}{T} \cdot \frac{\ln R}{1-\beta}, \tag{5}
\]

where \( t \) is the latest interval when the \( n^{th} \) flow is generated. \( M \) is the possible maximum value of \( C_{s,d,n} \) (e.g., if we use link utilization as the metric, the maximum value is 1), and \( R \) is the number of candidate routes. Equation (5) indicates that the difference of the cost of the algorithm \( E(\sum_{n=1}^{T} C_{s,d,n}) \) and that of the best strategy \( \min_k \sum_{n=1}^{T} r_t(q_t(L_{i,k}), q_t(L_{d,k}))/T \) can be bounded.

We next explain how to derive Eq. (5). We use Eq. (9) in Ref. [5] to derive Eq. (5). Equation (9) in Ref. [5] is \( L_{\text{Hedge}}(\beta) \leq \min_{l_{i,t}} \left[ \ln 1/(1-\beta^2) + \ln N \right] - \ln \left[ \frac{1-\beta}{\beta^2} \right] \cdot \frac{M}{T} \cdot \frac{\ln R}{1-\beta}, \)

where \( \min \sum_{n=1}^{T} r_t(q_t(L_{i,k}), q_t(L_{d,k})) \), and \( R \) in Eq. (5) corresponds to \( L_{\text{Hedge}}(\beta) \), \( \min l_{i,t} = \min \sum_{n=1}^{T} l_{i,t} \), and \( N \) in Eq. (9) in Ref. [5], respectively. Here, \( l_{i,t} \in \{0,1\} \). By using \( -\ln \beta \leq (1-\beta^2)/(2\beta) \) and \( \min \sum_{n=1}^{T} l_{i,t} \leq T \) because \( l_{i,t} \leq 1 \), Eq. (9) in Ref. [5] is simplified to Eq. (5).
rate between nodes $i$ and $j$ is proportional to $p_i \times p_j$, where $p_i$ is the population of node $i$ in a topology in Fig. 3; the population of each node is available in Ref. [20].

Each t-flow requests a file where file size is determined according to a truncated Pareto distribution whose probability density function $f(y)$ is

$$f(y) = k \times a^{(k+1)} \left( \frac{y^{k+1}}{u^{k+1}} - 1 \right),$$

where $k$ is the shape parameter, and $a$ and $u$ are the lower and upper bounds for file size, respectively. We set $k = 1.5$, $a = 4$ Mbytes, and $u = 137$ Mbytes so that average file size is 10 Mbytes.

Each flow has an access link speed with $A = 10$ Mbps, while each link capacity in a topology is set to $C = 155$ Mbps. To model TCP behaviors, we simulate that the max-min fair-share bandwidth is allocated to each flow when the number of active flows on a link exceeds $C/A$. That is, when a flow has a lower bandwidth requirement than the fair share bandwidth (due to access link speed limit) or the network assigns a lower bandwidth to the flow on another more congested link, the flow cannot use up its fair share bandwidth. In this case, the excess bandwidth is fairly split among all the other flows.

In the d-flow case, a flow has a constant rate of 10 [Mbps] and its duration is determined according to the same truncated Pareto distribution function as in the t-flow case.

In our simulation model, to be precise, t-flows or d-flows between nodes $i$ and $j$ are generated in both directions. That is, when a t-flow (or d-flow) is generated according to a Poisson process from nodes $i$ to $j$, another t-flow (or d-flow) is also generated from nodes $j$ to $i$ at the same time. Evaluating the case of asymmetric traffic patterns such as clients downloading the files from the server is for further study.

As for parameter setting, we used Eq. (7) to determine $\beta$ in our method. We set the number of candidate routes $R$ to $N_n - 1$, where $N_n$ is the number of physical nodes, because each source-destination nOL node pair has $N_n - 2$ relay routes and 1 direct route as candidate routes in the full-mesh topology. We set the number of flows $T$ to be the mean number of flows per node-pair generated over the time-length of a simulation run.

B. Evaluation results for t-flow

In this subsection, we evaluate a case where only t-flows exist. As application performance, we show the 5th percentile of flow throughputs under our method in Fig. 4. For comparison, we also show the results under “Selfish,” where each overlay node uses the average throughput measured between the overlay nodes and determines the best route, and “NoControl,” where no overlay routing is performed. In each topology, we offer the maximum acceptable traffic that NoControl can accept without continuous increase of the number of uncompleted file transfers. Figure 4 shows that our method can achieve almost the same high throughput as the access link speed. In contrast, Selfish and NoControl had degraded throughput due to some links becoming bottlenecked. In NoControl, this was because the route was determined by the minimum hop route in the physical network. As for Selfish, flows were routed to the same link, which caused the congestion.

As network resource utilization, we show the average of link utilization values for all links in Fig. 5, where “Optimal” means the case when we solve the minimum cost flow problem.

The optimal utilization is calculated as follows. We solve the minimum cost flow problem, i.e., “Given a network $G(V,E)$ with a nonnegative cost $c_{ij}$ and a nonnegative capacity $u_{ij}$ associated with every edge $(i, j) \in E$, and a specified pair of vertices (a source $s \in V$ and a destination $d \in V$) with traffic demand $D$, find a flow $f$ from $s$ to $t$ such that the sum of costs over all edges of a flow is minimized.” That is, the formulation is to find the amount of traffic $x_{ij}$ passing along edge $(i, j)$ that minimizes $\sum_{(i,j) \in E} c_{ij} x_{ij}$, subject to

$$\sum_{(s,j) \in E} x_{sj} - \sum_{(i,d) \in E} x_{id} = D, \quad 0 \leq x_{ij} \leq u_{ij}, \forall (i, j) \in E. \quad (8)$$

We set $u_{ij}$ to link capacity $C$. Because we want to minimize the overall traffic offered to the underlay network, we set $c_{ij}$ to 1. By using the primal-dual algorithm, we obtain the optimal results.

Figure 5 shows that our method can reduce the offered traffic compared with Selfish and achieves almost the same good performance as Optimal.

C. Case of coexistence of delay-sensitive and throughput-sensitive flows

We next evaluate a case where not only t-flows but also d-flows exist. The mean end-to-end delay of d-flows is evaluated as follows. We first calculate the time-average of the offered traffic of each link $i$ that is on the flow route over the flow duration. Then, by setting the average packet size to 1000 bytes and using the calculated offered traffic, we calculate the mean queueing delay of link $i$ by using the M/M/1 model, then we sum up the mean queueing delays over all the links on the flow route.

We evaluate the performance when the offered traffic is set to the maximum acceptable traffic volume by NoControl. The traffic mix ratio is set so that the mean number of d-flows is the same as that of t-flows. Figures 6 and 7 show the mean file transfer time of t-flows and mean end-to-end queueing delay of d-flows, respectively. From the graphs, we find that our method works well even when d-flows coexist with t-flows.
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2011 proceedings

M is the number of contents, and we set

throughput-sensitive flows (t-flows) ing delay of delay-sensitive flows (d-
when our overlay routing is and is flows) when our overlay routing is
and is not used.

D. Future work for evaluation for cacheable traffic

As a future work, we are evaluating the case when the traffic is cacheable. The model is as follows: In a network, there is only one original server that has all the contents. Content requests (flows) are generated according to a Poisson process whose arrival rate from node \( i \) is proportional to the population \( p_i \) of node \( i \) in a topology in Fig. 3, and flows are delivered in the same way as in the case of t-flows. We model the access frequency of content as a Zipf distribution. Specifically, the probability \( q(m) \) that a user requests a content whose popularity rank is \( m \) is determined by \( q(m) = c/m^{1-\theta} \), where \( c \) is the normalization constant to make \( \sum_{m=1}^{M} q(m) = 1 \), \( M \) is the number of contents, and we set \( \theta = 0.199 \) [22].

We set the total cache storage capacity \( K \) Gbyte. We determined the number of cache servers \( C \), each of which has the same storage capacity \( B (B = K/C) \), and the placement nodes so that we minimize the total amount of traffic passing across the network [21].

Content is cached to a cache server(s) as follows. If there is a node(s) with a cache server(s) on the route along which the content is delivered, we cache the content to the corresponding cache server(s) unless the storage space is fully utilized. If the storage space is fully utilized, the target content replaces another content by means of a cache-replacing algorithm, such as least recently used.

We have developed the above simulation model. We are evaluating the case of the cacheable traffic and investigating the impact of topology and parameters such as cache capacity on the performance of our method, which is for further study.

VI. CONCLUSION

We presented an architecture to control the traffic from various service overlay networks with an overlay network operated by an ISP or a carrier. Through the cooperation between overlay and underlay networks, we can improve the application performance as well as the resource utilization in network providers. We also presented an overlay routing operated by an ISP or a carrier, based on the Hedge algorithm, an online learning algorithm, which guarantees a theoretical bound for the obtained performance. Through simulation analysis for various network topologies, we showed that our overlay routing achieves good performance in terms of file transfer time and end-to-end delay as application performance criteria and efficient network utilization close to the optimal one obtained by solving the minimum cost flow problem.

Our future work will focus on how to extend our overlay architecture from the intra-domain to the inter-domain. We will also investigate how to allocate overlay nodes and to construct individual service overlay topologies, as this paper focused on the routing method. We will perform a more correct comparison between the proposed scheme and multiple selfish routing protocols running with different optimization metric (latency, bandwidth, etc.). Furthermore, we will investigate the effectiveness of our method combined with cache mechanism using simulation model in Section V-D.

REFERENCES

[22] H. Yu et al., “Understanding User Behavior in Large-Scale Video-on-Demand Systems,” ACM EuroSys 06.