Correlation-Aware Multimedia Content Distribution in Overlay Networks

Ying Zhu, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario M5S 3G4
Canada

ABSTRACT

We address the question: What is the best way to construct a mesh overlay topology for multimedia content distribution, such that the highest streaming rate can be achieved? We model overlay capacity correlations as linear capacity constraints (LCC) and propose a distributed algorithm that constructs an overlay mesh which incorporates heuristically inferred linear capacity constraints. Our simulations results confirm the accuracy of representing overlays using our LCC model and show the LCC-overlay achieving substantial improvement in achievable flow rate.

Keywords: Overlay networks, overlay construction, peer-to-peer networks, network protocol

1. INTRODUCTION

One of the most fundamental challenges in overlay content distribution of multimedia content is the high demand for available capacities in the underlying network. The recent advent of H.264/AVC content can easily command sustained streaming rates of over 1 Mbps. Due to the flexibility afforded by edge nodes (or *peers*), it is common practice to construct *mesh* overlay topologies to distribute multimedia content. As the most important form of such distribution is *streaming*, there has been a significant body of recent work in the area of overlay multimedia streaming. While most existing work focus on the optimization of streaming latencies, we believe that a critical Quality of Service parameter for any multimedia content distribution is the *sustainable flow rate* of the streaming sessions. Ideally, we prefer to achieve the highest streaming rates that the underlying network can sustain. The means to achieve such an objective is to construct the most suitable overlay topology for content distribution, which best exploits the available capacities in the underlying network.

Unfortunately, it is a challenge to construct such an overlay mesh topology, simply because we have minimal knowledge of the underlying IP network, especially with respect to its available capacity. As overlay links in the msh may unavoidably share underlying IP links (*e.g.*, the "last-mile" access link at the ISP), there exist *bottleneck links* in the underlying network, shared by overlay links, leading to deteriorated achievable flow rates in the mesh overlay.

Obviously, if we have complete knowledge of the underlying IP network topology, as well as their link capacities, it may be possible to design an optimal overlay topology. Such a *topology discovery* process, while noteworthy for theoretical research, is too expensive with respect to overhead in both probing traffic and time. A number of previous work require or propose protocols to acquire such knowledge. Other previous work adopted the restricted view that overlay links in a mesh topology are stand-alone and independent, and that their available capacities can be probed using point-to-point bandwidth probing techniques. We argue, instead, that overlay link capacities are inherently *correlated*, due to their sharing of available capacities in the underlying links. More specifically, an overlay link maps to an underlying path, and overlay links may map to paths that have shared links. The capacity of a common underlying link is shared by all the overlay links mapped to it: we call this overlay *link correlation*.

Thus, the question we would like to address in this paper is: Given the lack of complete knowledge of underlying topology, what is the best way to construct an overlay mesh topology for multimedia content distribution, which can achieve the highest possible sustained streaming rate, but with *minimal* probing overhead?

Towards this objective, we study the problem of constructing a *correlation-aware* overlay mesh topology with a distributed algorithm. In such an algorithm, link correlations are inferred with minimal probing. To design such an algorithm,

E-mail: {yz, bli}@eecg.toronto.edu

we propose an overlay model to accurately represent the overlay, and model link correlation with *linear inequality constraints* for overlay flow rates. We show that such linear constraints provide *succinct* and *sufficient* information to achieve modeling accuracy, as compared to complete knowledge of the underlying network capacities. We show that if such link correlations are not considered, it may incur grave deleterious effects on the sustainable flow rates of the topology. To infer link correlations and linear capacity constraints of an overlay, our algorithm utilizes both latency measurements and efficient bottleneck-sharing techniques. Our simulation results show that even a limited localized enhancement using capacity constraints is sufficient to notably improve the sustainable flow rate of the existing mesh construction strategy.

The remainder of this paper is organized as follows. We highlight related work in Section 2. Section 3 describes our overlay model of linear capacity constraints. In Sections 4 and 5, we present our distributed algorithm for mesh construction integrating capacity constraints, and discuss enhancing existing mesh strategies with constraints. Simulation results are presented in Section 6. Finally, we conclude in 7.

2. RELATED WORK

Some previous work (*e.g.*, Young *et al.*[1]) on overlay mesh construction adopted the view that overlay links are independent. Most commonly, the metrics of latency and bandwidth of a link are measured by unicast probing and assigned to overlay links as scalars. In contrast, in our previous work,[2] we argued that overlay links are not independent, and are inherently correlated. We have proposed the model of *linear capacity constraints*, but only studied overlay unicast problems in a largely theoretical study. In this paper, we investigate the problem of constructing an accurate overlay network in order to attain sustained high flow rates for multimedia content distribution.

Research has been active in recent years in peer-to-peer media streaming. As examples, a peer-to-peer streaming architecture is introduced in Zimmermann *et al.*[3] that significantly reduces end-to-end delay for interactive media streaming services. The problem of load distribution in multiple-path data streaming is studied in Abdouni *et al.*.[4] In,[5] the authors propose a peer-to-peer adaptive layered streaming framework. Their work is mainly on receiver-side mechanisms which adaptively control packet distribution among coding layers and which maximize overall quality while minimizing variations in playback quality in the presence of dynamics in available bandwidth. A number of papers, *e.g.*,,[6, 7] propose algorithms for encoded video streaming, but focus on problems from the encoding aspect such as segmentation for transmission and joining/leaving of layers.

The previous work most relevant to ours is PROMISE,[8] a peer-to-peer media streaming system that includes functions of peer lookup, aggregated streaming from multiple peers and dynamic adaptation to network conditions. The best sending peers are selected by a topology-aware selection technique. The sender-to-receiver paths are optimized based on a preliminary step of network tomography, which infers an approximate underlying topology. Our *correlation-aware* content distribution differs from PROMISE, in that we have relaxed the requirements of explicit discovery of underlying paths, by abstracting link correlation into linear capacity constraints — obtained through efficient end-to-end probing.

A significant amount of work has also been completed in multimedia content distribution, including overlay multicast,[9] content distribution[10] and multimedia streaming.[11] Among them, the topology-aware overlay construction algorithms by Ratnasamy *et al.*[12] are especially noteworthy: they incorporate more topological awareness into overlay construction. This work differs from ours in focusing exclusively on the latency metric, while our goal is to achieve high sustained flow rates.

3. OVERLAY NETWORK MODEL OF LINEAR CAPACITY CONSTRAINTS

We now present the model we propose for overlay networks to provide the service for correlation-aware multimedia content distribution. To facilitate our studies, we first make the assumptions that (1) The routing path in the underlying network between two end systems does not vary, and is decided by IP-layer routing protocols; and (2) When k overlay flows share the same underlying link with capacity b, every flow is entitled to a capacity of at most b/k.

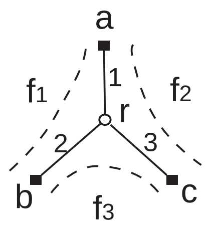
Overlay is so named because there is a network hierarchy of two layers: overlay is the higher layer; the underlying physical network forms the lower layer, henceforth also referred to as the underlay. The underlay is a graph $G_u = (S \cup R, E)$ whose set of nodes is a disjoint union of the set of routers R and the set of overlay nodes S. In theory, because every pair of end systems may form a virtual link, the overlay is a complete graph of the nodes S, let it be denoted by $K^{|S|}$. Let $M : (s,t) \in K^{|S|} \to P \subset E$ be a mapping from overlay links to paths in G_u. For every link between nodes s and t in $K^{|S|}$, $M(s,t) = (s, e_1, e_2, \ldots, e_l, t)$ maps it to a path from s to t through underlay edges $\{e_i : i = 1 \ldots l\} \subset E$. We assign a flow variable $f(s,t)$ to every link $(s,t) \in K^{|S|}$, representing its capacity. It follows then that if a set of overlay links $\{(s_i,t_i)\}_1^k$ maps to the same link $e \in E$, $\sum_{i=1}^k f(s_i,t_i)$ cannot exceed the capacity of e. Given the graph G_u and the set of overlay nodes S, it is straightforward to obtain the complete set of linear constraints for overlay link capacities.

Table 1. Common notations in the paper

Notation	*Definition*
S	set of overlay nodes
$\|S\|$	number of overlay nodes
R	set of routers in underlay
E	set of edges in underlay
G_u	$G_u = (R \cup S, E)$, underlay graph
$K^{\|S\|}$	complete (overlay) graph of $\|S\|$ nodes
m	$m = \|S\|(\|S\|-1)/2$, number of edges in $K^{\|S\|}$
s	source node in multicast
M	linear capacity constraints matrix ($\|E\|$-by-m)
x	$x = [x_1, \ldots, x_m]^T$, overlay flow variables vector
c	$c = [c_1, \ldots, c_{\|E\|}]^T$, capacity vector of edges in E

To illustrate, we give a naive example of an underlay graph consisting of only four nodes, as shown in Fig. 1, where $S = \{a,b,c\}$ and $R = \{r\}$, $E = \{(a,r),(b,r),(c,r)\}$. Capacity of an edge in E is as labeled in the figure. We let f_1, f_2, f_3 be flow variables for the three overlay edges, respectively, as labeled. The mapping of overlay edges to underlying paths is shown in the figure.

The edge (a,r) is traversed by two overlay edges $(a,b),(a,c)$, hence $f_1+f_2 \le w(a,r)$. Similarly, the linear constraints $f_1+f_3 \le w(b,r)$, $f_2+f_3 \le w(c,r)$ are obtained for the other two edges in E. The matrix form of the complete set of linear constraints is given next to the network graph in Fig. 1.



$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} \le \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Figure 1. A naive example.

Devising an algorithm for deriving a complete set of linear capacity constraints is therefore straightforward, given $G_u = (S \cup R, E)$, S and a mapping M of overlay edges in $K^{|S|}$ to paths in G_u. One such algorithm is to order the edges in E as $\{e_1, e_2, \ldots, e_{|E|}\}$ as well as the $|S|(|S|-1)/2$ (m) overlay edges $\{p_1, \ldots, p_m\}$. Matrix A is initialized to have all zero elements. For each p_i, $M(p_i)$ is the corresponding underlay path; set $A(j,i) = 1$ if and only if $e_j \in M(p_i)$. The complete set of constraints is thus $Mx \le c$, where $x = [x_1 x_2 \ldots x_m]^T$ is the vector of variables for overlay flows $p_1, \ldots, p_m$, and c is the vector of capacities for underlay links $e_1, \ldots, e_{|E|}$. The dimensions of M is $|E|$-by-m. We call M the constraint matrix, x the flow variable vector, and c the link capacity vector.

Theoretically and ideally, an $|S|$-node overlay is perfectly accurately defined by this model of the complete graph $K^{|S|}$ with the complete set of linear capacity constraints for the $|S|$ flow variables. The currently prevailing model of the overlay is as a network graph with no link correlations and the links labeled by numbers representing *independent* unicast latency and capacity. The independent model misrepresents the actual overlay metrics. Generally, previous work offset the detrimental misrepresentation by limiting the degree of an overlay node. Each node selects a limited number of incident links according to some selection rules, usually those that are good by certain metrics.

We examine a specific instance of such an overlay construction algorithm, let it be denoted by *OC*. In *OC*, a node selects d highest-capacity incident links (or neighbors).[*] This is a reasonable representative selection rule with respect to the objective of optimizing the flow rate. The following argument will be valid against any such ad hoc selection rule based on the independent overlay model.

To compare the overlay graph constructed by *OC* and the overlay defined by linear capacity constraints, we employ one of the most common multimedia content distribution topologies — the overlay multicast tree. We use a greedy algorithm *MT* to obtain the multicast tree with the objective of optimizing its flow rate: Given a source node s, the tree is initialized to $T = \{s\}$. The highest-capacity link with only one end-node in T is chosen and its other end-node is added to T. Ties are broken randomly.

The above greedy algorithm takes as input a graph with links having capacities as scalar weights. A slight modification is sufficient for the algorithm to take linear constraints of link flows as input. When deciding on a link to add to T, select the one that would have the highest capacity when getting its equal share of any shared underlay links with overlay links already in T. To be more precise: Let M, x, c denote the constraint matrix, overlay flow variable vector and underlay link capacity vector, respectively. Both M and c are input. Also let v_T be an m-by-1 vector (m is the number of overlay flow variables) such that $v_T(p) = 1$ if p is in T, otherwise it is 0. We use v_T^p to denote v_T with $v_T(p)$ changed from 0 to 1. Then, the following overlay link is selected to add to T: $p' = \arg\max_p\{\min\{\frac{c(i)}{k(i)} : \forall k(i) \neq 0\} : \forall v_T(p) = 0 \text{ and } k = Mv_T^p\}$; $v_T(p') = 1$.

Next we consider the simple example depicted in Fig. 2 — Fig.2(a) is a network G_u with $|R| = 4$ and $|S| = 4$ (4 routers, 4 overlay nodes), and Fig.2(b) is the complete overlay K^4 with the unicast capacity labeling the links. The overlay-to-underlay mapping is the obvious: $(A, B) \rightarrow (A, r_1, B)$ and similarly for (C, D); $(A, C) \rightarrow (A, r_2, r_3, C)$ and similarly for $(A, D), (B, D)$. Executing *MT* on K^4 yields the multicast tree T_{OC} in Fig. 2(c). Note that this is an optimal multicast tree if *only* K^4 is given. It is implicit that the number of neighbors selected is $d = 3$ in *OC* in this case; however, it is not hard to see that with $d = 2$ or $d = 1$, the same result ensues. Given the constraints

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{AB} \\ x_{AC} \\ x_{AD} \\ x_{BC} \\ x_{BD} \\ x_{CD} \end{pmatrix} \leq \begin{pmatrix} 2 \\ 3 \\ 2 \end{pmatrix} \tag{1}$$

MT will give the multicast tree T_{LC} in Fig. 2(d). It is easy to see that T_{OC} has actual flow rate 1 due to three flows sharing the link (r_2, r_3). Yet T_{LC} achieves flow rate 2.

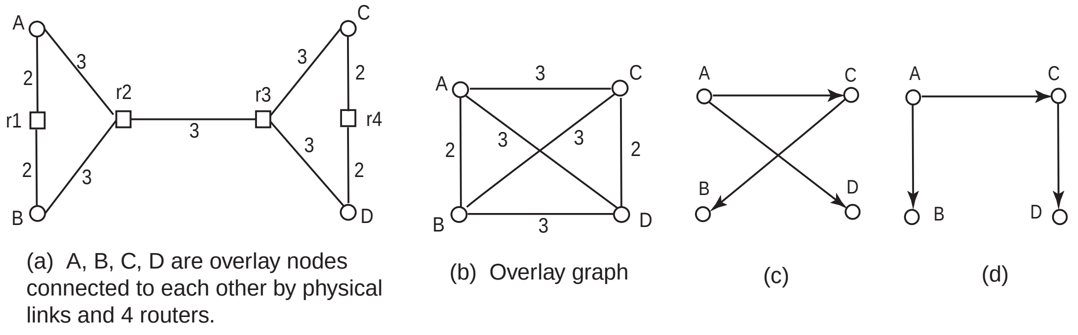

A
B
C
D
r1
r2
r3
r4
2
3
(a) A, B, C, D are overlay nodes connected to each other by physical links and 4 routers.
(b) Overlay graph
(c)
(d)

Figure 2. A simple example network showing the potential detrimental effect of the independent model of overlay.

Taking a cue from our simple example, we arrive at the following proposition in our previous paper[2] and it is included for completeness of this paper.

[*]Though fictitious, this is a slightly simpler version of the selection rule from[13] in which $d/2$ neighbors are selected from lowest latency ones and the other $d/2$ from highest capacity ones among randomly probed nodes.

Proposition 1: For any fixed value of $|S|$, there is a G_u such that the flow rate of an optimal multicast tree in any overlay graph (for any value of d) constructed by *OC* on top of G_u is asymptotically $1/|S|$ of the flow rate of a multicast tree obtained from flow constraints (M and c) derived from G_u.

Proof: Consider a generalized graph $G_u = (R \cup S, E)$ of the one in Fig. 2 with $|S|$ overlay nodes instead of only 4. The layout of G_u can be seen in Fig. 3(a). A single inter-router link in E connect $|S|/2$ overlay nodes with the other $|S|/2$ nodes. In the case of $|S|$ being odd, the nodes are partitioned $(|S|+1)/2$ and $(|S|-1)/2$, and the succeeding reasoning still holds. Any overlay graph constructed by *OC* will clearly include the $(\beta+\epsilon)$-link for every node. An optimal multicast tree in the *OC* graph must include only the $(\beta+\epsilon)$-links — one possible such tree is given in Fig. 3(b) — because otherwise its calculated flow rate would be $\beta < (\beta+\epsilon)$. However, the actual flow rate of this tree mapped to G_u is only $(\beta+\epsilon)/|S|$ since all $|S|$ links in the tree traverse the same inter-router $(\beta+\epsilon)$-link.

In the informed constraints overlay model, however, *MT* on the input of the overlay flow constraints, M and c, will form a multicast tree with flow rate β. With $(\beta-\epsilon)$ approaching 0, the *OC* tree asymptotically achieves $1/|S|$ of β. $\square$

Although at first glance, the network in Fig. 3 may appear pathological, it is worth noting that its topology resembles the scenario of a transatlantic link — real and inevitable in wide-area networks.

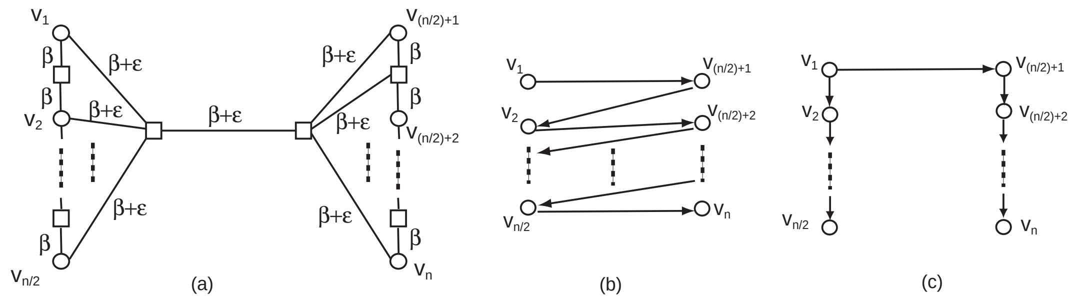


Figure 3. A $|S|$-overlay-node network showing the potential detrimental effect of the independent model of overlay.

4. LCC-NEIGHBORHOOD ALGORITHM DESIGN

The design objectives of the algorithm are: 1. Construct a general-purpose overlay mesh. 2. Every overlay node maintains information of linear capacity constraints. 3. Distributed, intelligent and limited (as opposed to exhaustive) inference of capacity constraints. We first describe the high-level design of the algorithm in the following section. In subsequent sections, every high-level component will be elucidated in detail.

4.1. High-level Overview

Every node randomly probes another node at regular intervals, maintaining and updating their stored measurement information. Both inter-node latency and capacity are measured. Every node also estimates its last-mile capacity[†]. Further discussion of this network measurement is deferred to Sec. 4.2.

Based on the 'distance' metric of latency, nodes distributedly group themselves into *neighborhoods* — each neighborhood contains nodes that are close to each other, while nodes from different neighborhoods are relatively far apart.

The rationale behind neighborhood grouping is the following proposition. In large-scale wide-area networks, latency (especially averaged over multiple probe results over a period of time to account for network fluctuations) is universally acknowledged and used as a measure of node proximity in a network.

Proposition 3: Assuming that IP layer routing does not deviate too far from shortest-path routing, if two nodes A and B have a high latency link between them, while node C and A have a low latency link, and node D and B have a low latency link. Then it implies that overlay links $A-C$ and $B-D$ do not share any bottleneck underlay links.

Proof: The scenario in the proposition is shown in Fig. 4. We prove by contradiction. Suppose the latency between A and B, l_{AB}, is high, but latencies l_{AC} and l_{BD} are low. Also suppose that $A-C$ and $B-D$ do share a bottleneck underlay

[†]The last-mile of a node is the capacity of its access link to the Internet. In our context, last-mile capacity is the constraint for the total capacity of all incoming and outgoing flows of a node.

link. Latency l_{CD} between node C and D must be low, because the path $C - e - D$ is all encompassed by segments of $C - A$ and $D - B$, thus $l_{CD} \leq l_{AC} + l_{BD}$, with the reasonable assumption that IP routing is not too far from shortest-path. It follows from the same reasoning that $l_{AB} \leq l_{AC} + l_{BD}$ is low as well. We reach a contradiction. $\square$

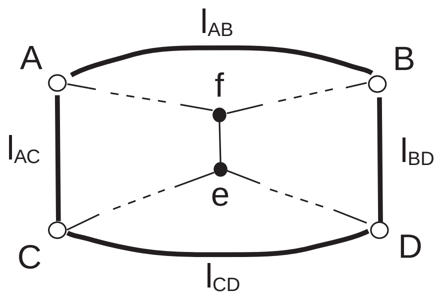


Figure 4. An example of the independence of links that are far apart with respect to latency. The bolder lines represent overlay links; the lighter lines represent underlay links.

Therefore, we conclude that it is not unreasonable to deduce independence of links from different neighborhoods, which implies that no capacity constraints exist for them. This reduces the need to probe capacity sharing between links from different neighborhoods and constraints within the neighborhoods can be probed and inferred independently of each other in parallel.

The reader will find in Sec. 4.3 details of how nodes distributedly group themselves into neighborhoods so that nearby nodes are grouped together. The neighborhoods can be viewed as *supernodes*; they also naturally form a complete graph. A subgraph is formed by every supernode selecting a number of adjacent supernodes to which incident superedges are maintained, we refer to it as *supergraph*.

Adjacent supernodes use capacity-sharing techniques to infer which edges in the supergraph share bottleneck capacity. (Detailed discussion about such techniques are deferred until Sec. 4.4.) These are the inter-neighborhood linear capacity constraints. They translate to the original overlay graph as follows. All inter-neighborhood flows inherit the constraints for corresponding supernodes. When the flow between two supernodes, say S_1 and S_2, is constrained by b, for all overlay nodes u in the neighborhood represented by S_1 and v in S_2, the sum of flows between nodes u, v is constrained by b. A more complicated example is the following. Nodes u_1, u_2 are in S_1 and v_1, v_2 are from S_2, and a constraint $f(S_1, S_2) \leq b$, the constraint $f(u_1, v_1) + f(u_1, v_2) + f(u_2, v_1) + f(u_2, v_2) \leq b$ is derived. Adding a fifth node w from a third supernode S_3 and the constraint $f(S_1, S_2) + f(S_1, S_3) \leq b'$, we induce the constraint $\sum_{i=1,2;j=1,2} f(u_i, v_j) + f(u_1, w) + f(u_2, w) \leq b'$.

For scalability reasons, the inter-neighborhood constraints are not explicitly stored by a node, which would include all possible flows from nodes in different neighborhoods. Instead, nodes exchange and store the respective supernode IDs of other nodes, and only keep the constraints of flows (if any) between their own supernode and other supernodes — which have far fewer variables. All inter-neighborhood constraints are effectively implicitly stored in this manner. For any set of overlay flows, any possible inter-neighborhood constraints that exist can be directly obtained from their associated supernode IDs and the stored supergraph flow constraints.

For any two adjacent supernodes in the supergraph, all the links between overlay nodes in different supernodes may be chosen to be in the overlay mesh, i.e., a node in one neighborhood may keep as many links to known nodes in the other neighborhood as computational efficiency dictates. A limit may easily be imposed should applications require it for the sake of efficiency: links with lower latency and higher capacity would be chosen. We remark that in theory, it is fine to keep all the links between neighborhoods that are adjacent in the supergraph — the constraints would disallow congestion due to inter-neighborhood flows, and better-performing data routes or topology may be found for various services due to the greater number of choices.

The full details of supergraph formation and inference of inter-neighborhood constraints are presented in Sec. 4.4.

Now it remains to consider overlay links within the same neighborhood — the intra-neighborhood links. In a neighborhood, the nodes first forms a spanning tree to ensure connectivity. Each node then selects d_{intra} incident links: $d_{intra}/2$ links have lowest latencies and the other $d_{intra}/2$ have highest capacities. Afterwards, a node waits for a randomly chosen period of time, and initiates probes for detecting capacity sharing among flows on its incident links. Every node thus obtains at most d_{intra} constraints with d_{intra} flow variables.

In the following sections, we proceed to present the details of all the individual algorithm components.

4.2. Unicast network measurement and last-mile estimation

All the nodes maintain and update whenever possible these pieces of information data: (1) a list of IP addresses of known nodes — node list; (2) latencies and capacities of nodes that have been measured; (3) IDs of neighborhoods associated with nodes; (4)last-mile capacity of nodes.

Latency and capacity are measured periodically for randomly chosen nodes from the node list, by the `ping` utility and a variant of the packet-bunch method.[14] Nodes update their node lists by periodically exchanging random subsets of them with randomly contacted nodes. A node may have to sacrifice completeness for scalability by keeping only a random subset of the node list when it grows too large.

For estimating last-mile capacity, we adopt a simple protocol which does not inject much extra network traffic. A node A maintains a small list of nodes that have the highest last-mile capacity as estimated by A thus far; it includes A. Initially, the last-mile list contains the nodes with which A has the highest-capacity links, and their last-mile estimates are their respective link capacity. The last-mile capacity of A is estimated to be the highest of these. Periodic exchanges and updates of last-mile lists are done with random nodes. Less frequently, node A also chooses a random node from the last-mile list to measure the link capacity, and its estimate of its own last-mile may be updated. The last-mile estimates are non-decreasing. If A has a high capacity, it will probe nodes with higher and higher capacity, and eventually estimate a comparable high capacity for itself. On the other hand, if A has low capacity, it is impossible to falsely estimate a higher one. Since a node's last-mile capacity is usually rather stable, once A detects that its own estimated last-mile has not changed in some time, A refrains from further active probing for a long time.

4.3. Grouping nodes into neighborhoods

We propose a low-overhead, distributed solution for grouping nearby nodes (with latency as the distance metric) into neighborhoods. Our solution requires only a small constant number of `ping`s and no landmark nodes (which is the case in[12]).

We consider classes of latencies rather than particular values. For instance, latencies may be divided into four classes: $(0, 10)$ms, $(10, 100)$ms, $(100, 200)$ms, $(200, \infty)$ms. If two nodes do not have a low latency link, e.g., latency is greater than 100ms, then they assume they are in different neighborhoods. Two nodes A and B (with relatively low latency link) determine whether they are in the same neighborhood in the following way. Node A chooses $g/4$ nodes whose latencies fall into each of the four latency classes — a total of g nodes chosen from relatively recently probed nodes. Node A sends these to B. Node B also sends g node IPs with latency classes to A. Now A checks its own measured latency classes for the g nodes *sent by* B, activating `ping` probes if necessary. Latency class of each of the g node as measured by A is compared with the class of that node sent by B. Node A conjectures that B is in the same neighborhood only if most of them coincide (i.e., a high fraction of them match, the fraction is a parameter that can be set). Node B proceeds in the same way. The two nodes are in the same neighborhood if both of them conjecture that. The procedure terminates when the size of a neighborhood reaches a certain threshold.

The effectiveness of our grouping scheme is verified by our simulation results, presented later in the paper, which reveal low overlay path latencies even when compared with minimum spanning tree algorithms that has minimization of latency as the main objective.

4.4. Inference of linear capacity constraints

With the nodes grouped, the neighborhoods can be logically shrunk into supernodes. A supernode is connected to another supernode through all the edges between nodes from (the neighborhoods corresponding to) the two respective supernodes. Our aim is a more sparsely connected supergraph. The question is: how do the supernodes select adjacent supernodes to keep in the supergraph?

In every neighborhood, an intra-neighborhood mesh — a sub-mesh of the overlay mesh being constructed — is built according to the protocol previously described at the end of Sec. 4.1. From each neighborhood S, h nodes are selected as the *anchor nodes* of S; the parameter h can be tuned for the tradeoff between overhead and precision. The criterion for an anchor node is high last-mile capacity. Every node has an estimate of its own last-mile; these estimates are used to choose the h highest ones. For the purpose of selecting anchors, it is straightforward to leverage the spanning tree infrastructure for control data dissemination.

A neighborhood S chooses another neighborhood T to be its adjacent supernode in the supergraph if the anchors in S have high capacity links from anchors in T. The degree limit of S in the supergraph is d_{inter}. The anchors in S records the sum of capacities of links from anchors in another supernode; the minimum latency is also recorded. Adjacent supernodes are chosen alternately from highest total capacity and lowest minimum latency, until S reaches d_{inter}.

The reasoning that led to the above heuristic of anchor selection is the following. The Internet consists of end system nodes that have vastly varying last-mile capacities. It does not help to have nodes with low last-mile capacity to probe for inter-neighborhood capacity, because all probes originating from and arriving at them will be automatically bounded by their low last-mile.

For each neighborhood S, after it has chosen its adjacent neighborhoods, the anchors in S utilize capacity sharing techniques to infer linear capacity constraints for all flows from anchors in adjacent supernodes to S. There have been several proposals of techniques that analyze multiple flows and detects shared bottlenecks among the flows, e.g.,.[15, 16] In particular, in,[15] Katabi *et al.* propose a technique to infer bottlenecks by minimizing the Renyi entropy (a generalized form of Shannon entropy) of the packet inter-arrivals. With only a small number of packets per flow, the method is able to partition flows into groups such that flows belonging to the same group share a common bottleneck whose capacity is also measured. They present Internet measurement results that demonstrate robust partitioning even under heavy cross traffic.

In practice, we only need to use 10 UDP messages of 8KB for each flow, since both d_{inter} and h are kept very small. We in fact were able to obtain good simulation results by setting d_{inter} to only 4 and h to 2. In this setting, inference of constraints for flows between S and all its adjacent neighborhoods costs 640KB.

Once flows are partitioned into groups that share bottlenecks and the corresponding bottleneck capacities are obtained, a set of linear capacity constraints for flows between S and its adjacent supernodes $\{S_i\}$ are inferred. The constraints inference is performed by anchors in every neighborhood. Thus, a set of constraints are determined for inter-neighborhood flows.

5. ENHANCING OVERLAY MESH CONSTRUCTION WITH LINEAR CAPACITY CONSTRAINTS

It may already be evident from the previous section that linear capacity constraints could be added to any overlay mesh construction scheme. We begin by briefly highlighting three previous mesh construction strategies: 1. *k Minimum spanning trees (k-MST)*: Nodes distributedly build k edge-disjoint minimum spanning trees, with every node keeping its best k links.[1] 2. *Short-long links (SL)*: Every node selects k links, by selecting $d/2$ of the shortest (lowest latency) links of which it has knowledge and selecting $d/2$ random (long) links.[12] 2. *Short-wide links (SW)*: Every node selects $k/2$ of the shortest links of which it has knowledge and $k/2$ of the widest (highest capacity) links from randomly probed nodes.[13]

Both *SL* and *SW* have node degree limit k, which is usually a small constant in practice. For instance, in,[13] *SW* sets k to be 8, so that every node has at most 8 incoming flows and 8 outgoing flows. Although in the simulations for *k-MST* in,[1] k varies only from 1 to 4, the algorithm does not impose a degree constraint. However, measures of adding a degree limit are discussed in.[1] Even without a degree limit, every node in *k-MST* has only k incoming edges, with $k \leq 4$.

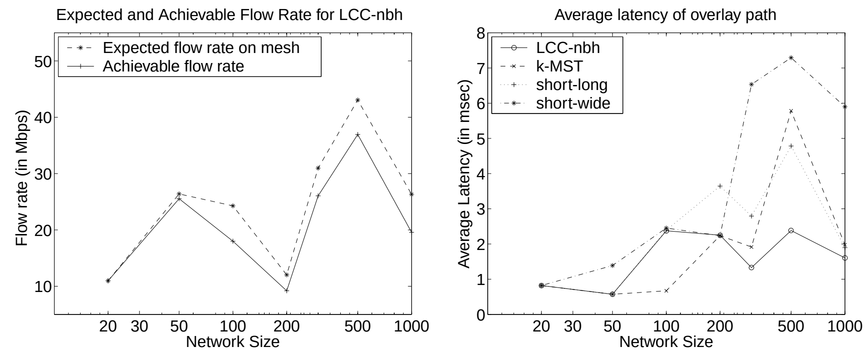
As it takes an exponential number of capacity-sharing tests to obtain a complete set of linear constraints, we experimented with various heuristics of adding inference of linear capacity constraints (LCC) to a general overlay mesh with a node degree limit. We were able to find one that is efficient, localized, distributed, and has comparable overhead as the mesh construction strategies themselves. It is also very simple. In the overlay mesh, nodes have at most k neighbors. Every node independently conducts probes for detection of bottleneck sharing among all flows on its incoming and outgoing edges. As discussed in the previous section, the bottleneck detection technique that we adopt actually partitions flows into groups such that flows in each group share a bottleneck and measures the respective bottleneck capacities.

Similar to probing parameters discussed in the previous section, we use 10 messages of size 8KB for each flow. There are at most k flows. Consider the *SW* mesh, every node has at most 16 incident links. Thus, the total cost of probing for bottleneck sharing is 1280KB. In,[13] in constructing an *SW* mesh, a capacity measurement test costs 480KB. In other words, if the bottleneck sharing probes are launched periodically with a sufficiently long period, it is not more costly than capacity measurement, which is a common periodic tool in overlay meshes.

We show our simulation results in Sec. 6.

6. SIMULATION RESULTS

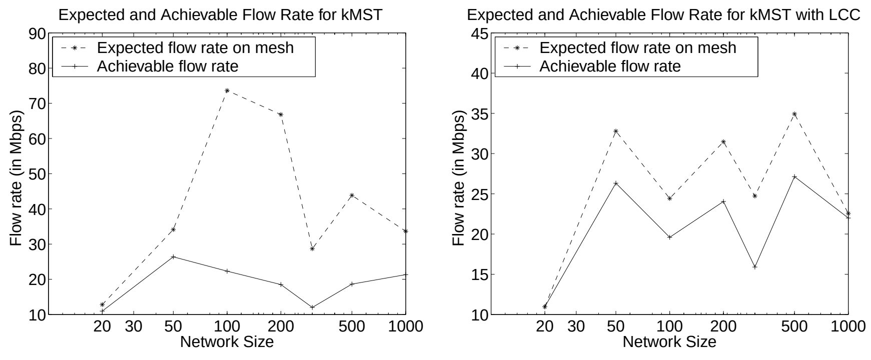
We now describe the results of our simulations on topologies generated by a power-law degree-based topology generator, BRITE.[17] Using both *C* and *Matlab*, we simulated and experimented with eight mesh algorithms altogether.



(a) Discrepancy between expected and achievable flow rate for maxflow on the LCC-nbh mesh

(b) Average latency of overlay path

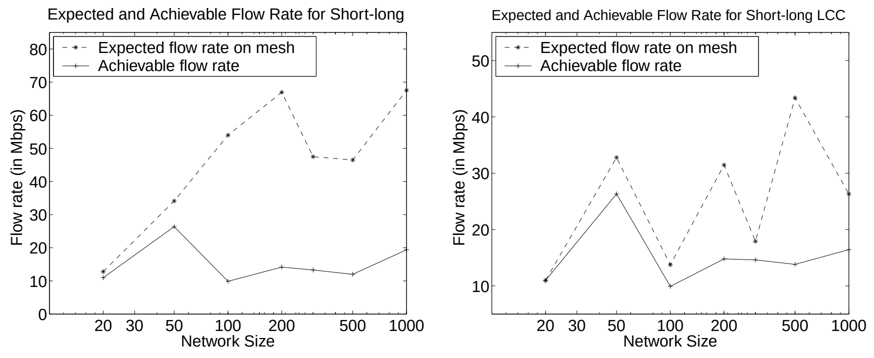
Figure 5. Discrepancy between expected and achievable flow rates; and average latency of overlay paths.



(a) k-*MST*

(b) k-*MST* with LCC

Figure 6. Discrepancy between expected and achievable flow rate for maxflow



(a) *Short-long*

(b) *Short-long* with LCC

Figure 7. Discrepancy between expected and achievable flow rate for maxflow

We list the algorithms with which we experimented: *LCC-nbh*: Nodes group by latency into neighborhoods, with inference of coarse inter-neighborhood constraints and localized independent intra-neighborhood constraints. k-*MST*: k minimum spanning tree mesh. *Short-long*: Short-long mesh. *Short-wide*: Short-wide mesh. k-*MST* with LCC: k-MST with localized linear capacity constraints. *Short-long* with LCC. *Short-wide* with LCC.

Overlay topologies range from 20 nodes and 1000 nodes. First, a larger router-level topology is generated using BRITE. From the router-level topology, nodes with the smallest degrees are selected to be overlay nodes. The *all-shortest-path* algorithm is executed to map overlay links to underlying paths in the router-level topology. The mapping of overlay links to underlay paths is stored for future reference. We implemented the mesh construction schemes listed above. From each of the overlay topologies (essentially complete graphs), a mesh is constructed using every algorithm. We choose to use maximum flow (maxflow) as a measure of the quality of the meshes with respect to flow rates that they can provide to receivers, in order that we may compare them. As a measurement of overlay quality, maxflow is a reasonable representative of multimedia content distribution applications. Moreover, its non-tree flow topology perfectly embodies the prevalent overlay network condition of multiple overlay links being simultaneously utilized, forming a topology that is not a tree.

We compute maxflow on the various overlay meshes constructed by the different algorithms. This maxflow is referred to as the expected maxflow by the meshes. Then we realize the overlay flows in the underlying network, using the mapping from overlay links to underlying paths. Whenever an underlying link is traversed by more than one overlay flow, the bandwidth of the link is allocated to the overlay flows by max-min fairness. After the overlay flows have been re-assigned their respective real achievable bandwidth, we execute maxflow on these overlay links again to obtain the overall achievable maxflow bandwidth.

The discrepancy between expected and real achievable maxflow bandwidth is plotted for every mesh construction algorithm. In Fig. 5(a), the two lines in the graph represent the expected bandwidth and achievable bandwidth, respectively, for our *LCC-nbh* algorithm that incorporates capacity constraints in the mesh. The algorithm is clearly effective in producing an accurate informed overlay mesh. This is evident from how closely the achievable bandwidth line follows the expected bandwidth line. The performance of the other three algorithms is substantially inferior. For all three, k-*MST* in Fig. 6(a), *Short-long* in Fig. 7(a) and *Short-wide* in Fig. 8(a), the achievable bandwidth is much (often wildly) lower than the expected bandwidth. This conclusively confirms our conjecture that ignoring the hidden capacity constraints will inevitably lead to a very inaccurate overlay mesh. We also define an error ratio: $(\text{expected} - \text{achievable})/\text{achievable}$. The error ratios are also plotted for all the algorithms in Fig. 9 and Fig. 10.

Now let us compare the discrepancy graphs for k-*MST* with and without LCC, shown in Fig. 6(b). With the constraints, the discrepancies between expected and achievable bandwidth shrink considerably. Note that this is with localized constraints that are obtain distributedly and are far from being complete. Evidently, significant improvement in accuracy can be achieved even for such limited information of capacity constraints. Similarly, as can be observed in Fig. 7(b) and Fig. 8(b), respectively, *Short-long* with LCC and *Short-wide* with LCC have much smaller discrepancy than *Short-long* and *Short-wide*, respectively, though the improvement is not as consistent as for k-*MST*.

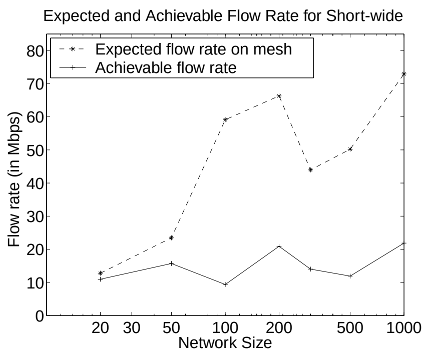
Next, we study the plots of achievable bandwidth in k-*MST* with LCC and that in just k-*MST*, the two lines representing them can be seen in Fig. 11(a). For larger network sizes of greater than 50, achievable bandwidth in k-*MST* with LCC is almost consistently higher by a significant amount. From this, we conclude that inaccurate overlay meshes cause inferior performance. It makes perfect sense: when the mesh is not accurate, an application algorithm cannot make informed choices in selecting routing paths and hence very likely will be misled into decisions that does not yield high performance. On the other hand, a number of limited localized capacity constraints help substantially in ameliorating the performance. However, in our experimentation with the *Short-long* with LCC, we discovered that this particular mesh with our simple heuristic of per-node constraint inference does not show clear increase in bandwidth, seen in Fig. 11(b). It is an indication that different heuristics for constraint inference have varying effect on different mesh strategies. Finally, we show the graph of average latency of overlay paths in Fig. 5(b). It can be seen that *LCC-nbh* has low latency compared to k-*MST*.

7. CONCLUDING REMARKS

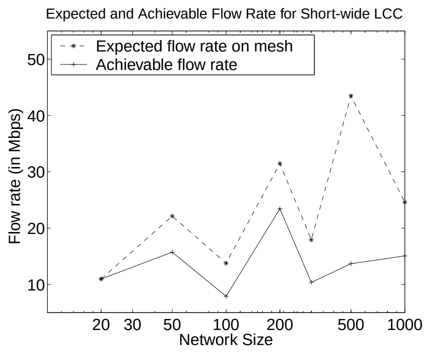
We considered the problem of constructing correlation-aware overlay meshes capable of maximizing the sustained flow rate in multimedia content distribution. An algorithm is proposed to construct overlay networks that are informed of underlying topological bandwidth constraints and hence accurate. Our simulation results verify the necessity of an accurate overlay mesh to ensure high flow rate.

REFERENCES

1. A. Young, J. Chen, Z. Ma, A. Krishnamurthy, L. Peterson, and R. Wang, "Overlay Mesh Construction Using Interleaved Spanning Trees," in *Proc. of INFOCOM*, 2004.

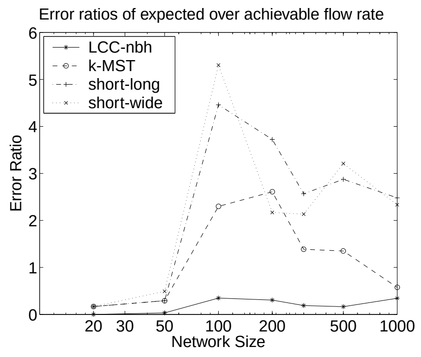


(a) *Short-wide*

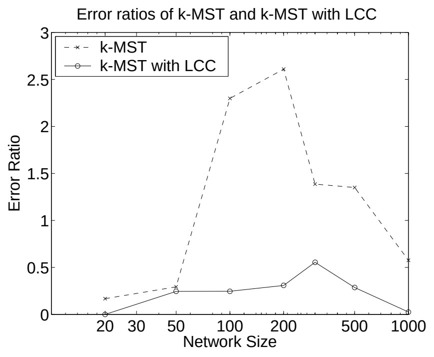


(b) *Short-wide* with LCC

Figure 8. Discrepancy between expected and achievable flow rate for maxflow

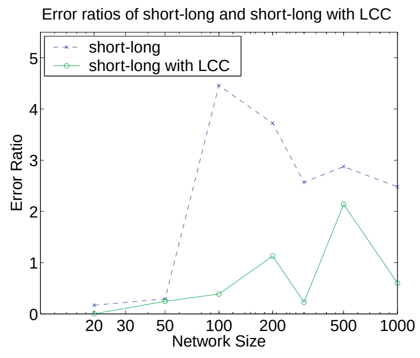


(a) *LCC-nbh* and the three mesh strategies

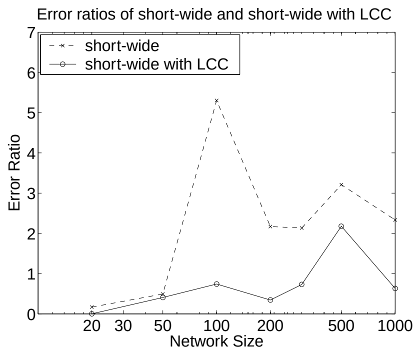


(b) k-*MST* and k-*MST* with LCC

Figure 9. Error ratio comparisons

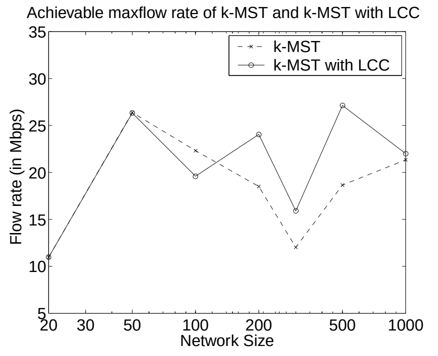


(a) *Short-long* and *Short-long* with LCC

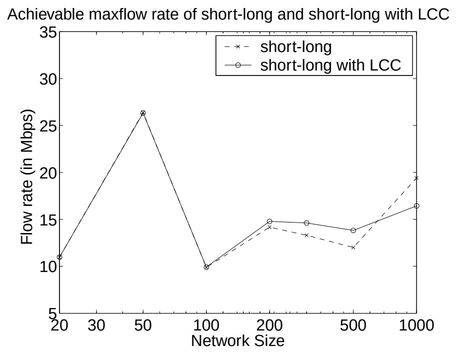


(b) *Short-wide* and *Short-wide* with LCC

Figure 10. Error ratio comparisons



(a) Achievable maxflow rate of k-*MST* and k-*MST* with LCC



(b) Achievable maxflow rate of *Short-long* and *Short-long* with LCC

Figure 11. Achievable maxflow rates of two vanilla and enhanced overlay construction schemes.

2. Y. Zhu and B. Li, "Overlay Networks with Linear Capacity Constraints," in *Proc. of the Thirteenth IEEE International Workshop on Quality of Service (IWQoS 2005)*, pp. 21–36, June 2005.
3. R. Zimmermann and L. S. Liu, "ACTIVE: Adaptive Low-latency Peer-to-Peer Streaming," in *Proc. of MMCN*, 2005.
4. B. Abdouni, W. C. Cheng, A. L. H. Chow, L. Golubchik, W.-J. Lee, and J. C. Lui, "Multi-path Streaming: Optimization and Evaluation," in *Proc. of MMCN*, 2005.
5. R. Rejaie and A. Ortega, "PALS: Peer-to-Peer Adaptive Layered Streaming," in *Proc. of NOSSDAV*, 2003.
6. C. Huang, R. Janakiraman, and L. Xu, "Loss-resilient On-demand Media Streaming Using Priority Encoding," in *Proc. of ACM Multimedia*, 2004.
7. L. Wu, R. Sharma, and B. Smith, "Thin Streams: An Architecture for Multicasting Layered Video," in *Proc. of NOSSDAV*, 1997.
8. M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: Peer-to-Peer Media Streaming Using CollectCast," in *Proc. of ACM Multimedia*, 2003.
9. Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," *IEEE Journal on Selected Areas in Communications* , pp. 1456–1471, October 2002.
10. J. Byers and J. Considine, "Informed Content Delivery Across Adaptive Overlay Networks," in *Proc. of ACM SIGCOMM*, August 2002.
11. V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking," in *Proc. of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2002)*, (Miami Beach, Florida), May 2002.
12. S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-Aware Overlay Construction and Server Selection," in *Proc. of the IEEE INFOCOM*, 2002.
13. K. Shen, "Structure Management for Scalable Overlay Service Construction," in *Proc. of NSDI*, 2004.
14. V. Paxson, "End-to-End Internet Packet Dynamics," in *Proc. of ACM SIGCOMM*, 1997.
15. D. Katabi and C. Blake, "Inferring Congestion Sharing and Path Characteristics from Packet Interarrival Times," tech. rep., Laboratory of Computer Science, Massachusetts Institute of Technology, 2001.
16. D. Rubenstein, J. Kurose, and D. Towsley, "Detecting Shared Congestion of Flows Via End-to-end Measurements," in *Proc. of ACM SIGMETRICS*, 2000.
17. A. Medina, A. Lakhina, I. Matta, and J. Byers, *BRITE: Boston University Representative Internet Topology Generator*, http://www.cs.bu.edu/brite.