Overlay Networks with Linear Capacity Constraints

Ying Zhu and Baochun Li, *Senior Member, IEEE*

Abstract—Overlay networks are virtual networks residing over the IP network; consequently, overlay links may share hidden lower level bottlenecks. Previous work has assumed an independent overlay model: a graph with independent link capacities. We introduce a model of overlays that incorporates correlated link capacities and linear capacity constraints (LCC) to formulate hidden shared bottlenecks; we refer to these as *LCC-overlays*. We define metrics to qualitatively measure overlay quality in terms of its accuracy (in representing the true network topology) and efficiency (that is, performance). Through analysis and simulations, we show that LCC-overlay is perfectly accurate and, hence, enjoys much higher efficiency than the inaccurate independent overlay. We discover that even a highly restricted LCC class—node-based LCC—yields near-optimal accuracy and significantly higher efficiency. We study two network flow problems in the context of LCC-graphs: widest path and maximum flow. We prove that Widest Path with LCC is NP-complete. We formulate Maximum Flow with LCC as a linear program and propose an efficient distributed algorithm to solve it. Based on the LCC model, we further study the problem of optimizing delay while still maintaining optimal or near-optimal bandwidth. We also outline a distributed algorithm to efficiently construct an overlay with node-based LCC.

Index Terms—Overlay networks, network protocols, algorithm/protocol design and analysis, network topology.

1 Introduction

The proliferation of research on overlay networks stems from their versatility, ease of deployment, and applicability in useful network services such as application-layer multicast [1], [2], media streaming, and content distribution [3]. Previous studies have uniformly taken the view of an overlay network as merely a weighted network graph; the nodes are end systems, the links are unicast connections, and the links are weighted by unicast delay and bandwidth. Overlay networks are therefore treated exactly as a flat single-level network in which the overlay links are independent. In particular, link capacities are independent of each other. This model is inaccurate as the overlay network encompasses two levels: a virtual network of end systems residing on top of an underlying IP network. An overlay link maps to a path, determined by the routing protocols, in the underlying network. When two or more overlay links map to paths that share an underlying link, the sum of the capacities of the overlay links are constrained by the capacity of the shared link, that is, these overlay links are *correlated* in capacity. This obvious but crucial observation leads us to conclude that an accurate model of overlay networks must include *link correlations*.

In this paper, we propose the model of overlay networks with linear capacity constraints (LCC). An LCC-overlay is a network graph in which the capacities of overlay links are represented by variables and link correlations are formulated as linear constraints of link capacities (that is, LCC). The LCC-overlay model is a succinct way to accurately represent the true network topology with all its link correlations, requiring only the addition of a set of LCC to the simple overlay graph.

We address the following questions: How do we qualitatively measure the quality of an overlay? Why do we prefer LCC-overlays instead of a simple network graph with independent links? Our analysis and simulations reveal the necessity of LCC-overlay in assuring the quality of overlay networks, and we introduce two qualitative metrics—accuracy and efficiency—to measure overlay quality. We also study a restricted class of LCC, node-based LCC, that is more efficient and of a distributed nature. Surprisingly, we find that, even with such restricted and incomplete LCC, the accuracy and efficiency are much better than overlays with no LCC and they are close to overlays with complete LCC. We propose a distributed algorithm for constructing an LCC-overlay based on node-based LCC.

We further study two network flow problems, widest path (that is, maximum-bandwidth single-path unicast) and maximum flow (that is, maximum-bandwidth multiple-path unicast) with the addition of LCC. Traditional algorithms cannot be used to solve them in a network graph with LCC. We show that widest path with LCC (WPC) is NP-complete. We formulate the problem of maximum flow with LCC as a linear program and propose an efficient algorithm for solving it.

Due to the importance of the end-to-end delay metric, we study an interesting variant of the problem of maximum flow with LCC by introducing the additional metric of delay: Instead of optimizing only bandwidth, simultaneously optimize both the bandwidth and the delay metric.

• Y. Zhu is with the Faculty of Business and Information Technology and the Faculty of Engineering and Applied Science, University of Ontario Institute of Technology (UOIT), 2000 Simcoe Street North, Oshawa, Ontario, L1H 7K4, Canada. E-mail: ying.zhu@uoit.ca.
• B. Li is with the Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, Ontario, M5S 3G4, Canada. E-mail: bli@eecg.toronto.edu.

Manuscript received 12 July 2006; revised 24 May 2007; accepted 5 June 2007; published online 27 June 2007.
Recommended for acceptance by X. Jia.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0186-0706.
Digital Object Identifier no. 10.1109/TPDS.2007.70726.

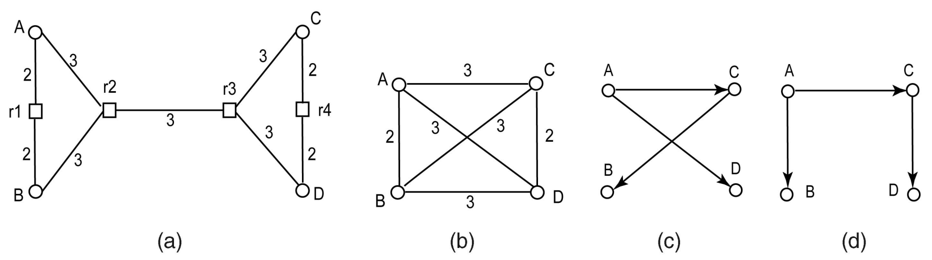


Fig. 1. A simple example of the detrimental effect that the independent model of overlay has on the overlay quality. (a) A, B, C, and D are overlay nodes connected to each other by physical links and four routers. (b) Overlay graph.

However, the objective of minimizing delay is frequently in conflict with the objective of maximizing bandwidth. Therefore, we focus on solving the problem of obtaining minimum or near-minimum delay while achieving maximum or near-maximum bandwidth. We formulate the problem as a linear program and propose an algorithm to solve it.

The remainder of the paper is organized as follows: Section 2 will introduce the concept of overlays with LCC, provide formal definitions of the LCC-overlay and the quality metrics, and show the necessity of LCC-overlay in ensuring high overlay quality through analysis and simulations. In Section 3, we present the problem of WPC and show that it is NP-complete. In Section 4, the problem of maximum flow with LCC is presented and formulated using linear programming; an efficient algorithm for solving it is proposed. The problem of optimizing both bandwidth and delay is studied and analyzed extensively through simulations in Sections 5, 6, and 7. Then, in Section 8, we outline an algorithm for constructing an LCC-overlay. Section 9 describes the related work, and Section 10 concludes the paper.

2 Overlay with LCC

In this section, we will define an overlay with LCC and two metrics for measuring overlay quality—accuracy and efficiency. We will further demonstrate through analysis and simulation that LCC are necessary for reliably ensuring high quality of overlay networks.

As a result of the two-level hierarchical structure, overlay links are not physical links; instead, they are virtual links that correspond to paths in the lower level network. We define *link correlation* as follows: Overlay links are correlated if they map to underlying paths that share one or more physical links. Link correlation is a fundamental property of overlay networks. Nevertheless, in the current prevailing overlay model of a graph in which each link is weighted by its unicast capacity, the underlying assumption is that overlay links are independent, with independent capacities being the measured unicast bandwidths. Suppose two overlay links both map to a bottleneck physical link of capacity c, hence, each has the unicast bandwidth c in the overlay graph; clearly, when these links are simultaneously used for data transmission, each one has a capacity of only $c/2$ rather than c. Thus, the independent overlay may be egregiously inaccurate in representing the network in reality.

We propose an overlay model that accurately represents the real network topology by using LCC to succinctly formulate link correlations. We refer to it as an *LCC-overlay*. Essentially, it is a regular overlay graph, except the links are not weighted by numbers; instead, the link capacities are variables and a set of LCC express the constraints placed on overlay links by shared bottlenecks. The formal definition of an overlay with LCC will be presented later in Section 2.2.

2.1 Worst-Case Analysis of Overlays with No LCC

For the purpose of illustration, we examine a simple example of a two-level network, as seen in Fig. 1a. The mapping of overlay links to physical paths is the obvious one in the graph. We adopt a simplified overlay construction algorithm, denoted by OC, that is nevertheless representative of such algorithms proposed previously under the independent overlay model. In OC, every node selects d neighbors to which it has links with the highest bandwidth.[1] With $d = 3$, the overlay graph for our example network is shown in Fig. 1b; it is not hard to see that the results we reach below hold for all feasible d, that is, $d = 2$ and $d = 1$. The highest bandwidth multicast tree for this overlay graph can be obtained, as presented in Fig. 1c. Let it be denoted as T_{OC}. Although the *predicted* bandwidth of T_{OC} according to the overlay graph is 3, the actual *achievable* bandwidth of T_{OC} is clearly only 1 because all three links in the tree share the physical link (r_2, r_3) with capacity 3.

In contrast, under the LCC-overlay model, capacities of overlay links are variables and link correlations can be captured by LCC. For instance, the four links (A, C), (A, D), (B, C), and (B, D) are correlated; hence, the sum of their capacities is constrained by the capacity of the physical link (r_2, r_3) shared by them, that is, $x_{AC} + x_{AD} + x_{BC} + x_{BD} \leq c(r_2, r_3)$. The LCC for the overlay graph in Fig. 1b are given below in matrix form:

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

The overlay graph together with the LCC form an LCC-overlay. For the LCC-overlay in our example, the highest

1. Though fictitious, this is only a slightly simpler variation of the neighbor selection rule in [4], where $d/2$ neighbors are selected from the lowest latency ones and the other $d/2$ from the highest bandwidth ones among randomly probed nodes.

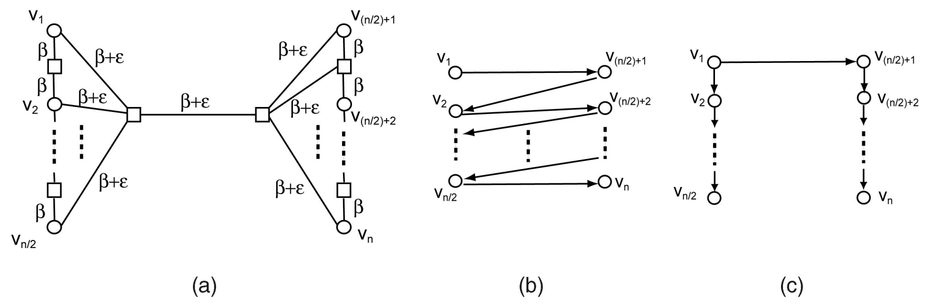


Fig. 2. A worst-case example of the poor quality of an overlay with no LCC.

bandwidth multicast tree is shown in Fig. 1d (obtained by a greedy algorithm that is a variation of the one for simple graphs, modified to take LCC into consideration). In this case, the predicted tree bandwidth is the same as the achievable bandwidth—both are 2.

Taking a cue from the above simple example, we arrive at the following:

Proposition. *For any fixed number of overlay nodes n, there exists a lower level network G such that the bandwidth of an optimal multicast tree in any overlay graph (for any value of d) constructed by OC on top of G is asymptotically $1/n$ of the bandwidth of an optimal multicast tree obtained in the LCC-overlay.*

Proof. Consider a generalized graph $G = (R \cup S, E)$ of the one in Fig. 1a with n overlay nodes, see Fig. 2a. A single physical link in the middle connect $n/2$ overlay nodes with the other $n/2$ nodes. In case of odd n, nodes are partitioned $(n+1)/2$ and $(n-1)/2$, and the succeeding reasoning still holds. Clearly, any overlay graph constructed by *OC* will contain the $(\beta + \epsilon)$-link for every link between the partitions, see Fig. 2b. An optimal multicast tree in the *OC* graph must include only the $(\beta + \epsilon)$-links because, otherwise, its predicted bandwidth would be suboptimal $(\beta < \beta + \epsilon)$. However, the actual achievable bandwidth of the tree mapped to G is only $(\beta + \epsilon)/n$ since all n links in the tree traverse the same interrouter $(\beta + \epsilon)$-link in the middle.

In the LCC-overlay, however, the optimal tree has bandwidth β, as shown in Fig. 2c. With ϵ approaching 0, the *OC* tree asymptotically achieves $1/n$ of β. □

An interesting corollary follows:

Corollary. *For any $\beta > 0$, a lower level graph G can be constructed such that 1) an optimal multicast tree in LCC-overlay has bandwidth β, and 2) as $n \to \infty$, the achievable bandwidth of an optimal multicast tree in the independent overlay (constructed by OC) asymptotically approaches 0 with decreasing ϵ.*

2.2 Formal Definitions of LCC-Overlay and Quality of Overlay

Our analysis above reveals that the worst case of an independent overlay with no LCC (No-LCC overlay) is that the optimal-bandwidth topology predicted in the overlay has only the achievable bandwidth of close to zero. From this egregious example, we observe that the extreme poor performance of the independent overlay is a consequence of its *inaccuracy* in representing the true network topology. The LCC-overlay, on the other hand, represents the network with perfect accuracy and, hence, achieves the optimal bandwidth.

Two questions now arise naturally: 1) How do we quantitatively measure the accuracy and the performance of achievable bandwidth or, in short, the quality of overlay networks? 2) How does the quality (that is, accuracy and performance) of LCC-overlays compare with that of No-LCC overlays in realistic network topologies?

Before we directly address these questions, we must first formally define the LCC-overlay and the metrics to measure overlay quality. We will also make the notion of an overlay flow with its predicted flow rate (that is, bandwidth) and its achievable bandwidth in the low-level network more precise.

The two-level hierarchy of an overlay network can be formulated as consisting of

- a low-level (IP) graph $G = (V, E)$; each low-level link $e \in E$ has a capacity of $c(e) \geq 0$;
- a high-level (overlay) graph $\widehat{G} = (\widehat{V}, \widehat{E})$, where $\widehat{V} \subset V$; and
- a mapping P of every overlay edge $(\widehat{v}_1, \widehat{v}_2) \in \widehat{E}$ to a low-level path $P(\widehat{v}_1, \widehat{v}_2) \subset G$ from $\widehat{v}_1$ to $\widehat{v}_2$.

The formulation of capacity constraints in the overlay graph $\widehat{G}$ is where LCC-overlay departs from No-LCC overlay. The No-LCC overlay is a pair $(\widehat{G}, \widehat{c})$, where $\widehat{c}$ is a capacity function such that each link $\widehat{e} \in \widehat{E}$ has a nonnegative capacity $\widehat{c}(\widehat{e}) \geq 0$. The LCC-overlay is defined as follows:

Definition 1 (LCC-overlay). *The* LCC-overlay *is a triplet $(\widehat{G}, C, b)$, where*

- *the capacity of each link $\widehat{e}$ in $\widehat{G}$ is a variable $x_{\widehat{e}}$ and*
- *(C, b) represents a set of m LCC $Cx \leq b$:*
  - *C is a 0-1 coefficient matrix of size $m \times |\widehat{E}|$,*
  - *x is the $|\widehat{E}| \times 1$ vector of link capacity variables, and*
  - *$b \in \mathrm{R}^m$ is the capacity vector.*

Each row i in (C, b) is a constraint of the form $\sum_{\widehat{e}:C(i,\widehat{e})=1} x_{\widehat{e}} \leq b(i)$.

A flow f from s to t in $\widehat{G}$ is an assignment of bandwidth to every link in $\widehat{E}$ subject to capacity constraints and flow conservation (that is, for every node except s and t, the total incoming bandwidth is equal to

```
for each $e \in E$
   use max-min fairness to allocate $c(e)$
     among $\{\widehat{e} : e \in P(\widehat{e})$ and $f(\widehat{e}) > 0\}$,
   let each allocation be denoted by $\gamma_f^e(\widehat{e})$
   for each $\widehat{e} \in \widehat{E}$
     if $f(\widehat{e}) > 0$
       $\gamma_f(\widehat{e}) \leftarrow \min\{\gamma_f^e(\widehat{e}) : e \in P(\widehat{e})\}$
     else
       $\gamma_f(\widehat{e}) \leftarrow 0$
   $\sigma_G(f) \leftarrow$ maximum-flow in $(\widehat{G}, \gamma_f)$
  $|\sigma_G(f)| \leftarrow$ bandwidth of $\sigma_G(f)$
```

Fig. 3. The procedure of determining $\sigma_G(f)$.

the total outgoing bandwidth). The value of the flow, $|f|$, is the total outgoing bandwidth of s.

In a No-LCC overlay $(\widehat{G}, \widehat{c})$, it is not always possible to achieve the predicted bandwidth $|f|$, as illustrated in the examples above. Links in $\widehat{G}$ may share bottlenecks in G, which impose more stringent constraints on their capacities than $\widehat{c}$, which assumes no correlation whatsoever.

We denote the *actual achievable flow* of $f \subset \widehat{G}$ in the low-level graph G by $\sigma_G(f)$ and the *actual achievable bandwidth* of f by $|\sigma_G(f)|$. We now describe the procedure for obtaining these.

Let f be a flow from node A to node C in the No-LCC overlay shown in Fig. 1b, with $f(A, C) = 3$, $f(A, B) = 2$, $f(B, C) = 2$, hence, $|f| = 5$. The low-level graph $G = (V, E)$ is shown in Fig. 1a. Supposing low-level link (r_2, r_3) is in $P(A, C) \cap P(B, C)$, then the true capacity of overlay links (A, C) and (B, C) in f is a fair share of the bottleneck capacity, denoted by $\gamma_f(A, C) = \gamma_f(B, C) = c(r_2, r_3)/2$. For link (A, B), $P(A, B) = \{(A, r_1), (r_1, B)\}$, thus $\gamma_f(A, B) = f(A, B)$. Using the true capacities of these three links with respect to f, a maximum flow from A to C can be obtained. This is the actual achievable flow of f, $\sigma_G(f)$, in which a flow of 1.5 is assigned to all three links and $|\sigma_G(f)| = 3$ is the achievable bandwidth of f.

In general, given G and a flow $f \subset \widehat{G}$, the procedure of determining $\sigma_G(f)$ is shown in Fig. 3. Note that this procedure only finds the achievable flow rate for a given overlay flow f (*not* the maximal achievable flow rate for the given overlay network) assuming that the overlay nodes try to send data at the flow rates assigned to the respective links in f. In this context, it is assumed that there is no knowledge of the underlying physical network.

We introduce two metrics for measuring the quality of an overlay network: *accuracy* and *efficiency*. We first define accuracy and efficiency with respect to a maximum flow f in the overlay. Accuracy is the predicted flow rate, $|f|$, divided by the achievable bandwidth of f, $|\sigma_G(f)|$. It essentially measures the degree to which the overlay overestimates a maximum flow; an accuracy value of 1 indicates perfect accuracy. Efficiency is the achievable bandwidth of f divided by the low-level maximum flow bandwidth; it measures how good an overlay maximum flow performs in comparison with the maximum flow in the low-level network where there are no path constraints. Note that the low-level maximum flow is the absolute optimal and cannot be attained on the overlay. The formal definitions are given as follows:

Definition 2 (Accuracy). *Accuracy of a maximum flow f in overlay network $\widehat{G}$ residing over G is*

$$\alpha_{\widehat{G}}^{f} = \frac{|\text{maximum flow } f \subset \widehat{G}|}{|\sigma_G(f)|}. \tag{2}$$

Definition 3 (Efficiency). *Efficiency of a maximum flow f in overlay network $\widehat{G}$ residing over G is*

$$\varepsilon_{\widehat{G}}^{f} = \frac{|\sigma_G(f)|}{|\text{maximum flow } \bar{f} \subset G|}. \tag{3}$$

It may be the case that some f have perfect accuracy and efficiency because they avoid low-level bottlenecks by mere chance. Therefore, the overall accuracy and efficiency of an overlay are better measured by taking the average of accuracy and efficiency over all possible maximum flows.

Definition 4 (Accuracy and Efficiency of Overlay). *Accuracy of an overlay $\widehat{G}$ is the* mean *of*

$$\{\alpha_{\widehat{G}}^{f} : s\text{-}t \text{ maximum flow } f, \forall s, t\}.$$

Efficiency of an overlay $\widehat{G}$ is the mean *of*

$$\{\varepsilon_{\widehat{G}}^{f} : s\text{-}t \text{ maximum flow } f, \forall s, t\}.$$

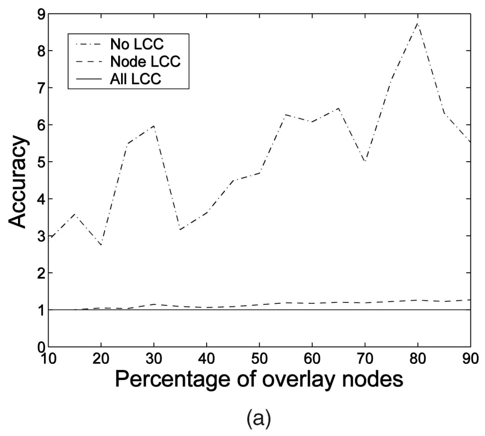
2.3 Comparing the Quality of No-LCC Overlay and LCC-Overlay in Realistic Internet-Like Topologies

In practical terms, to discover a complete set of LCC will incur high complexity and cost and will also require centralized operations. Motivated by this, we consider a restricted class of LCC: *node-based LCC*. A node-based LCC contains only capacity variables of links that are adjacent to a single node. In other words, every node can independently and distributedly discover its own set of node-based LCC by only probing its adjacent links. Therefore, we simulate three types of overlays: No-LCC, All-LCC, and Node-LCC.

Through simulations with realistic network topologies, we compare the quality of all three types of overlays using the accuracy and efficiency metrics defined above. For this purpose, we use an Internet topology generator, BRITE [5], which is based on power-law degree distributions.[2]

First, we compare the accuracy and efficiency of the three overlays with various overlay sizes relative to the low-level network size. We fix the number of low-level nodes to 100 and vary the number of overlay nodes from 10 to 90; the data for accuracy and efficiency are averaged over numerous maximum flows with randomly selected source and destination nodes. In Fig. 4a, accuracy is plotted against the ratio of overlay over low-level size. The All-LCC overlay always achieves its predicted maximum flows because it has all the bottleneck information in its complete LCC. Thus, All-LCC maintains a constant perfect accuracy of 1.

2. A seminal paper [6] by Faloutsos et al. revealed that degree distribution in the Internet is a power law. Another previous study in [7] provided evidence that degree-based topology generators, BRITE and a few others, model the Internet topology quite accurately.



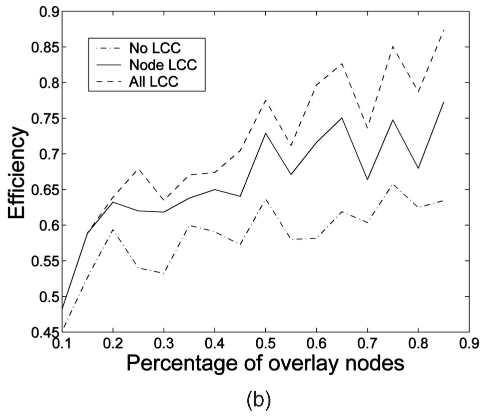
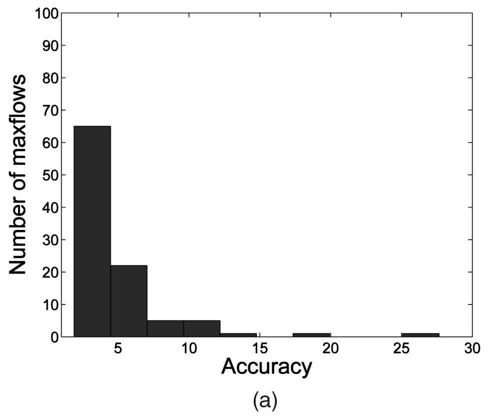


Fig. 4. Overlay quality versus ratio of overlay size to low-level size. (a) Accuracy of No-LCC, All-LCC, and Node-LCC overlays versus ratio of overlay size over low-level network size. (b) Efficiency of No-LCC, All-LCC, and Node-LCC overlays versus ratio of overlay size over low-level network size.

As the number of overlay nodes increases in a constant low-level network size, the accuracy of Node-LCC only deviates negligibly from 1, meaning its predicted maximum flows can (almost) always be achieved. No-LCC fares much worse with much higher values for the accuracy metric, which indicate that it is overly optimistic in predicting maximum flow values that cannot actually be achieved, and what is achieved by these maximum flows are substantially lower than predicted.

Fig. 4b shows the efficiency versus overlay-to-low-level ratio for the three overlays. The top curve is All-LCC with the highest efficiency of all three, as expected, since it possesses all bottleneck information—also for this reason, All-LCC has the optimal overlay efficiency, that is, higher efficiency cannot be achieved by only using overlay links. The surprise here is how closely the Node-LCC efficiency curve follows that of All-LCC for all overlay ratios less than 65 percent. In reality, overlay networks in the Internet are much smaller in size compared to the low-level IP network; therefore, Node-LCC has near-optimal overlay efficiency for realistic overlays. For most of the overlay ratios, No-LCC has significantly lower efficiency than both All-LCC and Node-LCC. It should be noted that No-LCC efficiency is not as poor as its accuracy, relative to the two LCC. This can be explained by the fact that No-LCC heavily overestimates (as indicated by its accuracy values) link capacities and, thus,



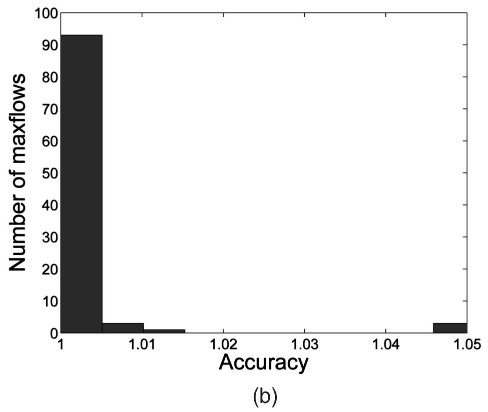


Fig. 5. Accuracy distributions for (a) No-LCC and (b) Node-LCC with the fixed ratio 30 percent of overlay to low-level size.

overloads the low-level links, causing them to be used to full capacity and thereby benefiting the efficiency. However, overloading low-level links has the disadvantage that other links are not utilized (or underutilized) because it was not foreseen that they were needed. This is why No-LCC is still significantly less efficient than Node-LCC.

Next, we evaluate the accuracy and efficiency of maximum flows with a fixed overlay-to-low-level ratio of 30 percent. The distributions of accuracy over 100 maximum flows for No-LCC and Node-LCC are given in Figs. 5a and 5b, respectively. As already mentioned above, effectively all Node-LCC maximum flows have perfect accuracy, whereas well over a third of No-LCC maximum flows have extremely poor accuracy (values greater than or equal to 5).

The distributions of efficiency over maximum flows are more interesting. In No-LCC, shown in Fig. 6a, over half of the maximum flows are less than 60 percent efficient and only 15 percent of them are 100 percent efficient. The distribution is quite different for All-LCC, seen in Fig. 6c, in which a much higher number (almost 25 percent) of maximum flows has 100 percent efficiency and over half of them have efficiencies higher than 70 percent. The Node-LCC distribution in Fig. 6b looks almost the same as All-LCC, with only slight differences. It has the same number of maximum flows with 100 percent efficiency and just about half of them have efficiency higher than 70 percent.

The coinciding of Node-LCC efficiency with All-LCC efficiency is confirmed in their cumulative distributions in Fig. 6d, where the two curves are almost the same. In this

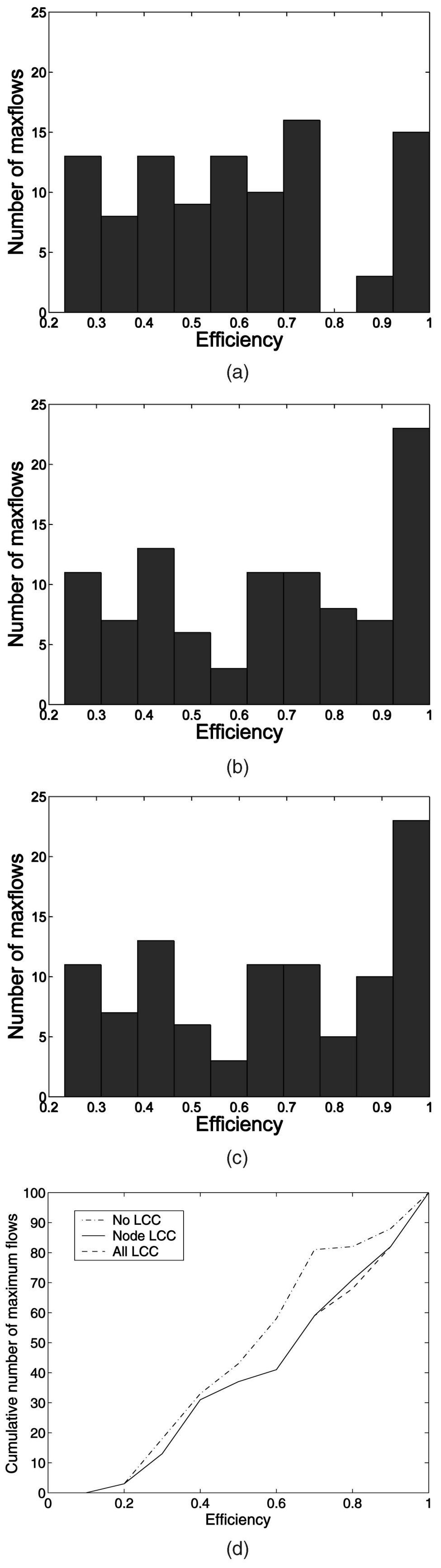


Fig. 6. Efficiency distributions of (a) No-LCC, (b) Node-LCC, and (c) All-LCC and (d) their cumulative distributions for a fixed ratio of 30 percent of overlay to low-level size.

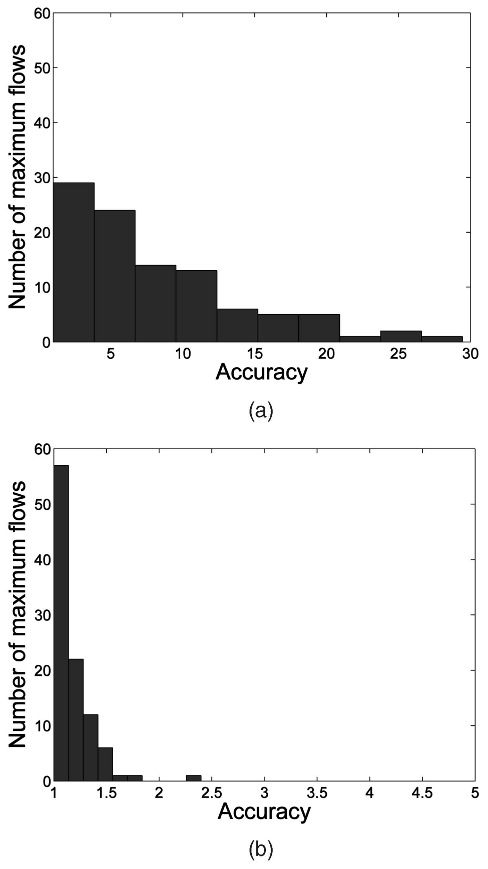


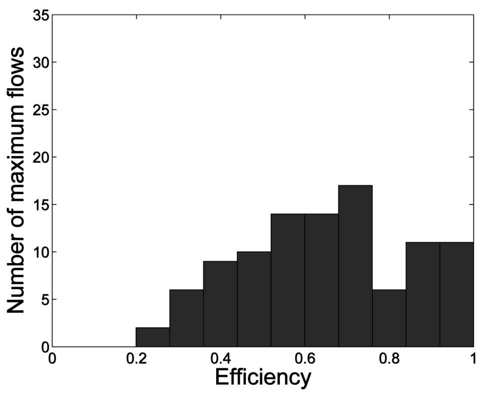
Fig. 7. Accuracy distributions for (a) No-LCC and (b) Node-LCC for network size 500.

graph, the observations made above can be seen more clearly. Most of the All-LCC and Node-LCC flows have higher efficiency, whereas most flows in No-LCC have lower efficiency.

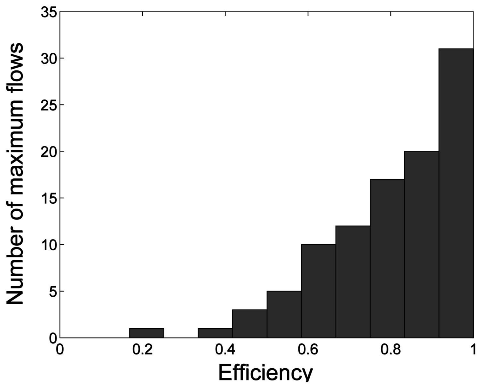
We examine the impact of larger network sizes on accuracy and efficiency by increasing the network size to 500 nodes and keeping the ratio of overlay to network size at 30 percent. Fig. 7 shows the accuracy distributions of No-LCC and Node-LCC. No-LCC accuracy is much worse than for the previous smaller network size. However, the increased network size causes only a tiny change in Node-LCC accuracy, which stays near 1.

Now, we proceed to compare the efficiency distributions. The efficiency distribution for All-LCC, given in Fig. 8c, shows extremely high efficiency for almost all the maximum flows sampled. Most of them have very high efficiency and a majority of flows are 100 percent efficient. All-LCC efficiency has significantly improved for increased network size. The reason, we conjecture, is that the low-level maximum flows have to travel longer paths in the larger network; thus, they are more similar to the paths that overlay flows map to, which means that both overlay and low-level maximum flows encounter many of the same bottlenecks.

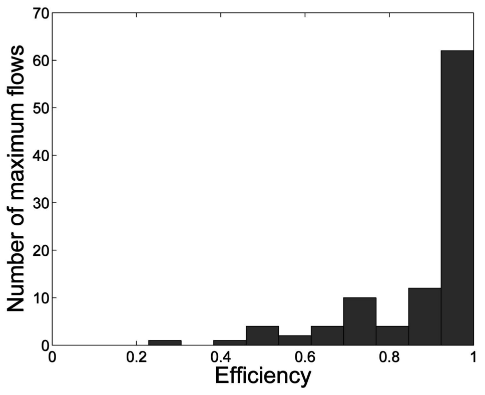
The same reasoning explains the improved efficiency for Node-LCC in this larger network; Fig. 8b shows its efficiency distribution. It is still similar in shape to All-LCC though with more differences than in the smaller network.



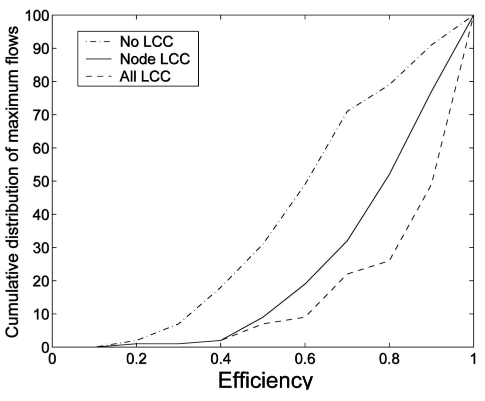
(a)



(b)



(c)



(d)

Fig. 8. Efficiency distributions for (a) No-LCC, (b) Node-LCC, and (c) All-LCC and (d) their cumulative distributions for network size 500.

On the other hand, as can be seen in Fig. 8a, No-LCC efficiency is even more inferior compared to Node-LCC (and of course All-LCC) than in the smaller network case.

The cumulative distribution graph in Fig. 8d illustrates that, while Node-LCC efficiency is no longer almost the same as All-LCC, its gap from All-LCC is still smaller than the gap between itself and No-LCC. In Node-LCC, most of the maximum flows have high efficiency. Moreover, it should be noted that Node-LCC is, just like All-LCC, much more efficient for the larger network size than for the smaller network size. We conclude that increasing network size causes significant deterioration in quality of No-LCC overlays but actually significantly improves the quality of All-LCC and Node-LCC overlays.

3 WPC Is NP-Complete

The LCC-overlay is an entirely different type of network graph than traditional network graphs. The algorithms for various network flow problems do not work in the LCC-graph. In this section, we consider the problem of WPC, that is, finding the highest bandwidth unicast path from a source s to a destination t. Widest path can be solved by a variation on Dijkstra's shortest path algorithm, however, this algorithm does not in general find the widest path in an LCC-graph. It is easy to see why: When a widest path is found based on independent link capacities, it may contain links that are constrained by an LCC; therefore, the path may not attain its predicted width (bandwidth) and its actual width is smaller than some other path.

We begin by formulating the problem of the WPC. We are given an LCC-graph $\{G = (V, E), C, b\}$, as defined above in Section 2.2.

The width of a path $p = \langle e_1, e_2, \ldots, e_k \rangle \subset G$, $w(p)$, is defined as

$$\begin{aligned} &\text{maximize} && x_{e_1} \\ &\text{subject to} && x_{e_j} = 0, \forall e_j \notin p, \\ & && Cx \leq b, \\ &\text{and} && x_{e_1} = x_{e_2} = \ldots = x_{e_k}. \end{aligned}$$

In the above, setting x_{e_i} equal for all $e_i \in p$ and minimizing one of them, say, x_{e_1}, is equivalent to maximizing the minimum of x_{e_i} subject to the other constraints. In practice, the above maximization problem can be solved by first assigning 1 to x_{e_i}, $\forall e_i \in p$ and 0 to the remaining variables. Multiply C by the assigned x and multiply the product by a vector of 1s (to sum the rows of the product) to obtain r. Divide b by r (if an element of r is 0, the division gives ∞) and take the minimum of these elements.

We define the widest path weight from u to v as $\omega(u, v) = \max\{w(p) : u \overset{p}{\leadsto} v\}$ if there is a path, and otherwise, it is 0. The widest path from u to v is any path p such that $w(p) = \omega(u, v)$.

We define the widest path with capacity constraints problem as a decision problem:

WPC

- *INSTANCE*. An LCC-graph (G, C, b), where $G = (V, E)$ and (C, b) are a set of LCC, specified s and t, a positive integer $K \leq \max\{b_i\}$.

- *QUESTION*. Is there a directed path p from s to t whose width is no less than K?

Theorem. *WPC is NP-complete.*

Proof. WPC is in NP because a nondeterministic algorithm need only guess a subset of E and check in polynomial time whether these edges form a path p with $w(p) \geq K$.

We transform the Path with Forbidden Pairs (PFP) [8] to WPC. The PFP problem is defined as follows: INSTANCE: Directed graph $G = (V, E)$, specified vertices $s, t \in V$, collection $F = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ of pairs of vertices from V. QUESTION: Is there a directed path from s to t in G that contains at most one vertex from each pair in F?

Let G, s, t, F be any instance of PFP. We must construct a graph $G' = (V', E')$, $s, t \in V'$, a set of LCC $Cx \leq b$ for edges in E', and a positive integer $K \leq \max_i\{b_i\}$ such that G' has a directed path from s to t of width no less than K if and only if there exists a directed path from s to t in G that contains at most one vertex from each pair in F.

Any vertex v that is in V and not in any pair in F remains unchanged in V'. Any edge $e \in E$ that is not incident to a vertex in F is added to E' without change. For every vertex u that appears in F, we replace it with two vertices u', u'' and we call a directed edge e_u from u' to u'' u's replacement edge. For every directed edge $e = (v, u) \in E$ that enters u, an edge $e' = (v, u')$ is added to E'; similarly, for every edge $e = (u, v) \in E$ that exits u, we add $e' = (u'', v)$.

Now, we form the LCC. Each edge $e \in E'$, except the edges that replaced vertices in F, gives rise to a one-variable constraint $x_e \leq 1$. For each pair of vertices $(a, b) \in F$ that have two replacement edges e_a and e_b in G', respectively, we form a two-variable constraint $x_{e_a} + x_{e_b} \leq 1$.

Finally, we set $K = 1$. Clearly, the construction can be accomplished in polynomial time.

Suppose that there exists a directed path p from s to t in G containing at most one vertex from each pair in F. A corresponding path p' can always be obtained in G' by simply substituting all p's constituent vertices that appear in F by their replacement edges in G'. If such a replacement edge e_a corresponding to a pair $(a, b) \in F$ is in p', then, obviously, $e_b \notin p$, that is, setting $x_{e_a} = 1$ and $x_{e_b} = 0$ gives e_a width 1 while conforming to the capacity constraint $x_{e_a} + x_{e_b} \leq 1$. If neither edges from a two-variable constraint appear in p', they can simply be set to have width 0. Thus, all two-variable constraints are satisfied. Any edge that is not a replacement edge for a vertex in F can be assigned a width of 1 without violating its corresponding one-variable constraint; therefore, all the one-variable constraints are satisfied. Since all edges in p' are assigned a width of 1, p' is the desired path for WPC in G'.

Conversely, let p' be an s–t path in G' satisfying all the constraints and having width no less than 1. The width of p' being no smaller than 1 implies that, in order to satisfy every two-variable constraint, if $e_a \in p'$, then $x_{e_a} = 1$, $x_{e_b} = 0$, otherwise, vice versa. In short, at most one edge from any two-variable constraint appears in p'. Collapsing p' to a path $p \in G$ by shrinking the replacement edges into their corresponding vertices, it is obvious that p satisfies the PFP condition. □

Even though the WPC problem is NP-complete, we discovered through simulations that the widest paths obtained without considering LCC (but only using independent link capacities) are able to achieve an actual bandwidth that is optimal or extremely close to optimal. The reason, we believe, is that it is highly unlikely for links in a single path to correlate heavily, which means that actual achievable bandwidth (width) is the same as or very close to the predicted bandwidth. The lesson here is that the traditional widest path algorithm suffices for realistic LCC-overlays. In general, however, the WPC problem—with considerations of all possible pathological cases—is still NP-complete.

4 Maximum Flow with LCC

In this section, we study the problem of maximum flow in an LCC graph. The traditional maximum flow algorithms such as Ford-Fulkerson and Push-Relabel cannot solve the maximum flow with LCC problem (MFC). We first formulate the problem as a linear program and then propose an algorithm for it based on Lagrangian relaxation and existing algorithms for minimum cost flow.

Maximum Flow with LCC Problem (MFC):

- *Input*. $\widehat{G} = (\widehat{V}, \widehat{E}), C, b$.
- *Output*. A flow $f \subset \widehat{G}$ satisfying LCC constraints (C, b).
- *Goal*. Maximize $|f|$.

Like the maximum flow problem, the MFC problem can be viewed naturally as a linear program. A variable v is used to indicate the total flow out of s and into t. In the flow conservation constraint, A is the node-arc adjacency matrix for $\widehat{G}$,[3] and h is a vector with a 0 for every node except $h(s) = -1$ and $h(t) = 1$:

$$\begin{aligned}
&\text{Maximize} && v \\
&\text{subject to} && \\
& && Af + hv = 0 \\
& && Cf \leq b \\
& && f \geq 0.
\end{aligned}$$

The MFC linear program can be solved by general linear programming algorithms, such as the simplex method. However, due to their general nature, they may lack flexibility and may not be as efficient as algorithms that are more specific and tailored to the problem. We propose an alternative solution of the MFC problem using Lagrangian relaxation and an existing combinatorial algorithm for solving the subproblem of minimum cost flow. With the characteristics of the MFC problem, the minimum cost flow algorithm we chose has the lowest complexity. Our simulation result presented later also shows that the

3. Rows are nodes; columns are edges; for each directed edge $e = (i \rightarrow j)$, $A(i, e) = 1$, $A(j, e) = -1$, otherwise, entries of A are zero.

Lagrangian relaxation technique converges relatively quickly.

Note that the MFC linear program only differs from the generic maximum flow linear program in having LCC $Cf \leq b$ as the inequality constraint instead of $f \leq b$. MFC can be seen as a generalized maximum flow problem; maximum flow is a special case of MFC with the identity matrix as C.

With that observation, we modify the linear program slightly to reveal even more clearly the embedded maximum flow structure in the MFC problem. We do this by sieving (uncorrelated) link capacity constraints from (C, b): For each link e, add the constraint $f(e) \leq b_l(e)$, where $b_l(e) = \min\{b(j) : C(j, e) = 1\}$, that is, minimize over all constraints in C involving $f(e)$. It is easy to see that the additional $f \leq b_l$ constraints do not change the feasible flow region; therefore, the new linear program is equivalent to the original one. For convenience in subsequent manipulation, the objective function and the equality constraint are expressed in a different form:

$$z^* = \text{Minimize} \qquad -v, \tag{4}$$

subject to

$$Af + hv = 0, \tag{5}$$

$$f \leq b_l, \tag{6}$$

$$Cf \leq b, \tag{7}$$

$$f \geq 0. \tag{8}$$

It is now evident that MFC is a maximum flow problem with some additional constraints (that is, the LCC) in (8); thus, we can adopt the decomposition solution strategy to exploit its underlying network structures for which efficient algorithms have already been developed.

We apply the solution method of Lagrangian relaxation [9] to the MFC problem by associating nonnegative Lagrange multipliers $\mu = [\mu_i]_1^m$ with the LCC constraints in 8 and creating the following Lagrangian subproblem:

$$L(\mu) = \min \quad -v + \mu(Cf - b), \tag{9}$$

subject to

$$Af + hv = 0, \qquad f \leq b_l, \qquad f \geq 0. \tag{10}$$

For any given vector μ of the Lagrangian multipliers, the value $L(\mu)$ of the Lagrangian function is a lower bound on the optimal objective function value $z^* = \min -v$ of the original problem (4). Hence, to obtain the best possible lower bound, we need to solve the Lagrangian multiplier problem:

$$L^* = \max_{\mu \geq 0} L(\mu). \tag{11}$$

Note that, for the our Lagrangian subproblem (9), for any fixed value of Lagrangian multipliers μ, $L(\mu)$ can be found by solving a minimum cost flow problem. A polynomial-time minimum cost flow algorithm is the cost scaling algorithm, with a running time of $O(n^3 \log(n \cdot K))$, where n is the number of nodes and K is the upper bound on all the coefficients in the objective function. Since the objective coefficients are 1 or -1, the time complexity in this case is $O(n^3 \log(n))$. We choose the cost scaling algorithm precisely because its running time depends neither on m (that is, number of LCC or rows in C), nor on U (upper bound on values in b_l).[4] The values of m and U may be rather large, whereas C is a constant in this specific minimum cost flow problem.

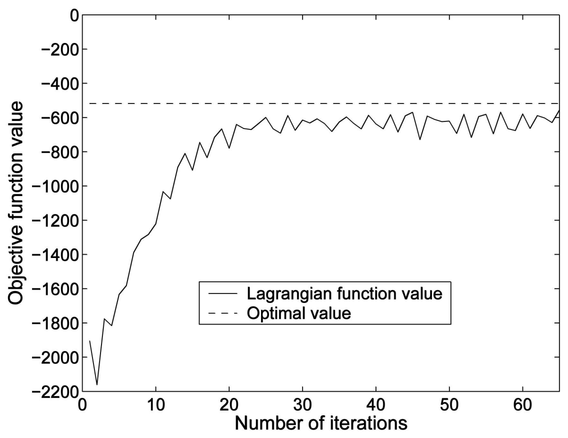


Fig. 9. Shows the convergence of Lagrangian function values (or Lagrangian subproblem solutions) $L(\mu)$ (in Problem 9) to a value near to the true optimal value z^* (in Problem 4), after a relatively small number of iterations.

Now that we can solve the Lagrangian subproblem for any specific μ, we can solve the Lagrangian multiplier problem (11) using the subgradient optimization technique (chosen because the Lagrangian function may not be differentiable for some values of μ). It is an iterative procedure: begin with an initial choice μ^0 of Lagrangian multipliers; the subsequent updated values μ^k are determined by $\mu^{k+1} = [\mu^k + \theta_k(Cx^k - b)]^+$. Here, the notation $[.]^+$ means taking the maximum of 0 and each component of the vector; x^k is a solution to the Lagrangian subproblem when $\mu = \mu^k$; θ_k is the step length at the kth iteration. The step length is selected to be a widely used heuristic,

$$\theta = \frac{\lambda_k(UB - L(\mu^k))}{\parallel Cx^k - b \parallel^2},$$

where $0 < \lambda_k < 2$, and UB is any upper bound on the optimal value of (4).[5]

We show in Fig. 9 that, for MFC in a simulated network in which 30 percent of the nodes are overlay nodes, the Lagrangian function values converge to a near optimal value in around 65 iterations.

5 Two-Metric Optimization

Aside from sustainable flow rate, end-to-end delay is also a critically important Quality-of-Service metric for multimedia content distribution. Thus, we have focused

4. All the other polynomial-time minimum cost flow algorithms described in [9] include either m or U or both.

5. It should be noted that sometimes there may be a gap between the optimal Lagrangian multiplier objective function value and the optimal value for the original problem, wherein the branch and bound method can be used to overcome the gap. We do not go into the details here.

exclusively on the optimization of a single metric: flow rate. However, end-to-end delay is often a metric of interest to be constrained or minimized; for instance, real-time streaming of video or audio is delay sensitive. Even non-real-time multimedia content distribution will certainly prefer to minimize delay while maintaining optimal rate. In this section, we take delay into consideration as an additional metric to be optimized. More specifically, we try to simultaneously minimize end-to-end delay and maximize flow rate and to find a trade-off between the two possibly conflicting objectives.

We must first clarify what exactly it is that we are trying to minimize, that is, define the formula of end-to-end delay that will be used in the optimization. The basic definition of end-to-end delay is the time it takes for a packet to be transmitted from source to destination. Note that, given a flow f from s to t, it can be decomposed into m single-path subflows $\{p_i\}_{i=1}^{m}$, such that f is a disjoint union of them, and each p_i traverses only a single path from s to t. Since quality of service is best measured by receiver perception, the delay of f should be defined as the maximum or the average of the delays of p_is. Given only f, it is not easy to decompose it and find or even estimate the maximum of single-path subflow delays. Fortunately, it *is* easy to obtain the total delay, and by minimizing the total delay, it is equivalent to minimizing the average single-path subflow delay. Hence, we study the problem of simultaneously maximizing the flow rate and minimizing the average path delay.

Since the goal is to find flows with high rates and low delays, we call them *Wide-Short* flows. We give the linear programming formulation for finding Wide-Short flows:

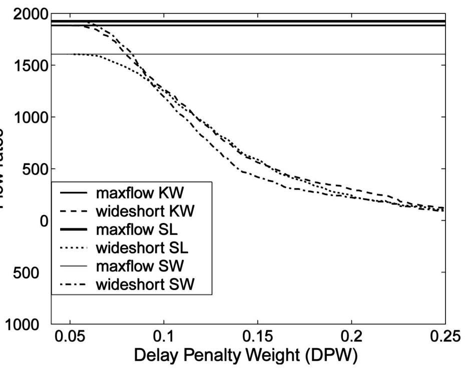
$$\begin{aligned}&\text{Maximize} && v - \alpha \cdot \textstyle\sum_{i:\hat{E}} d(i)f(i)\\ &\text{subject to} && \\ & && Af + hv = 0,\\ & && Cf \le b,\\ & && f \ge 0.\end{aligned}$$

As in the case of maximizing only the flow rate, A is the incidence matrix, C is the matrix for LCC, b is the capacity vector, and f is the vector of link flow variables. The difference is in the objective function. In addition to maximizing the total flow variable v, we also minimize $\alpha \cdot (\text{total delay})$. We refer to the parameter α as the delay penalty weight (DPW). Note that, in the objective function, a solution with higher flow rate is preferred, whereas the solution is also penalized for higher delay and α is the weight of the penalty based on how much the total delay is permitted to affect the objective function value.

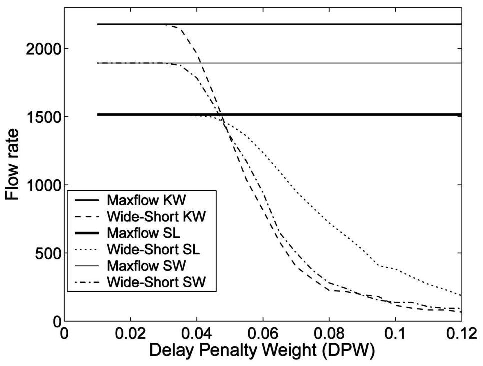
6 Delay Penalty Weight (DPW) in Wide-Short Flows

To evaluate the flow rates and delays of wide-short flows, we conduct experiments by simulations with *C* and *Matlab* on realistic topologies generated by, again, the power-law degree-based topology generator, BRITE.

We selected three reasonable strategies for constructing overlay meshes and implemented them in our simulations.



(a)



(b)

Fig. 10. (a) $\text{Network size} = 300$. (b) $\text{Network size} = 3,000$.

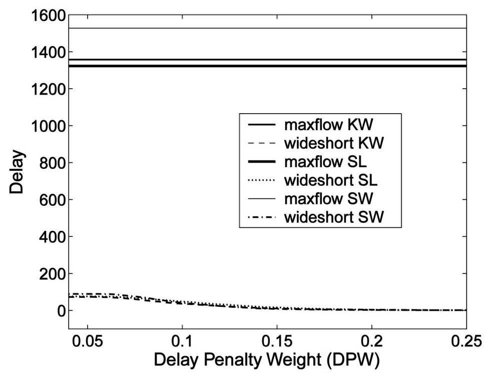
In the simulations, the mesh construction algorithms differ mainly in their neighbor selection protocols:

1. k-widest (KW). k neighbors with which the node has the highest capacity adjacent links.
2. Short long (SL). $k/2$ lowest delay adjacent links and $k/2$ randomly selected links. This was proposed in [10].
3. Short wide (SW). $k/2$ lowest delay adjacent links and $k/2$ highest capacity links. This was proposed in [4].

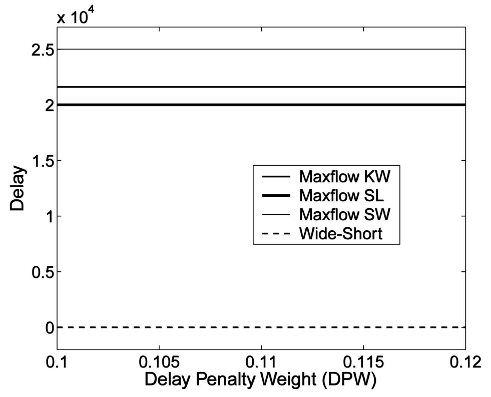
Each of the above three overlay meshes is constructed with LCC. In our simulations, the Wide-Short flows are obtained in all three types of LCC overlay meshes.

We first experimented with various values of the DPW α in the linear program. With network size 300 and 33 percent of the nodes being overlay nodes, α (DPW) is varied from 0.04 to 0.25. For each α (DPW) value, 100 pairs of source and destination nodes are randomly chosen, and thus, 100 maxflows and wide-short flows are obtained. This is done for the three types of LCC overlay meshes: KW, SL, and SW.

The maxflow and wide-short flow rates, averaged over the 100 runs, are plotted against the increasing DPW values in Fig. 10a. The average maxflow rates obviously do not change for different DPW values. For DPW values less than 0.06, the wide-short flow rates match maxflows. The three curves for wide-short flow rates (on KW, SL, and SW overlays) follow each other closely as they decrease over increasing DPW values. This indicates that, despite differences in neighbor selection rules in overlay mesh construction, the behavior of wide-short flow rates, as DPW varies, is essentially the same.

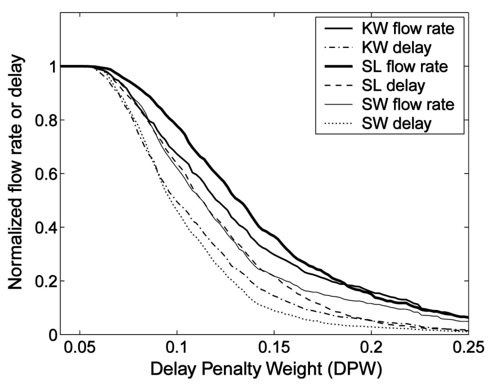


(a)

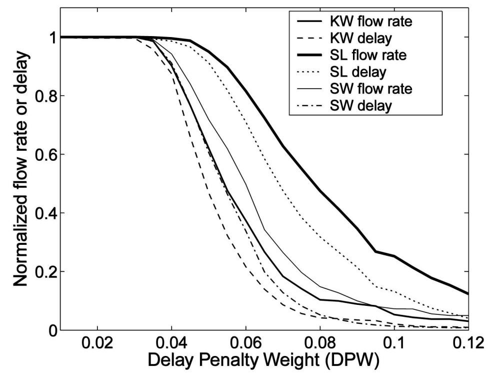


(b)

Fig. 11. (a) Network size $= 300$. (b) Network size $= 3,000$.



(a)



(b)

Fig. 12. (a) Network size $= 300$. (b) Network size $= 3,000$.

In Fig. 11a, the delays of maxflows and wide-short flows are shown. The delay of wide-short flows are far smaller than that of maxflows, going from almost two orders smaller to three orders smaller as DPW increases.

To see more clearly the trade-off between maximizing flow rate and minimizing delay, we show both wide-short flow rate and delay in the same graph. Since flow rate and delay have different units, it is not possible to compare them directly with their absolute values. Rather, we are interested in how much and how fast the decrease in flow rate is compared to the decrease in delay, as DPW increases. Therefore, for each of the three overlays, both flow rate and delay values are normalized by means of dividing each by their respective highest value, that is, dividing by the values when DPW is the smallest. A normalized flow rate is the fraction of maxflow rate that is achieved by wide-short flow. A normalized delay is the delay in proportion to the delay of wide-short flow at the smallest DPW (that is, α) value (in this experiment, 0.04). With the reference points of maxflow rate and delay at the smallest DPW in this experiment, the relative improvements in flow rate and delay may be compared with each other.

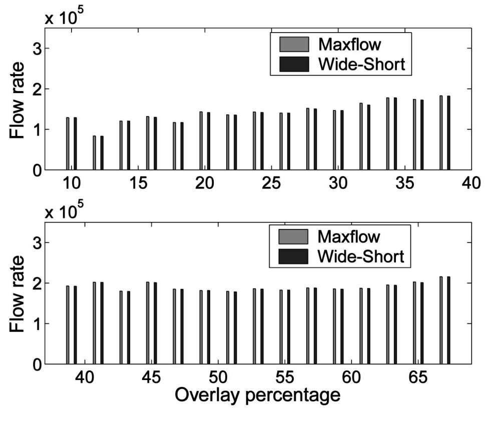
The normalized flow rate and delay of wide-short flows on the three overlays are shown in Fig. 12a. As usual, the results are average values over 100 runs with randomly selected source-destination pairs. Looking at the SL overlay, the flow rate curve is well above the delay curve, which is the promising sign that flow rates do not degrade nearly as quickly as the improvement in delay. At a DPW value in the middle of roughly 0.12, the wide-short flow rate is around 70 percent of the maximum, whereas its delay is already only 50 percent of the starting point. Almost the exact same trade-off relationship exists in KW and SW overlays. Once again, different overlays appear to exhibit the same behavior.

The same graphs for flow rates, delay, and normalized rates and delay, over increasing values of DPW, are plotted for a much larger network size of 3,000 in Figs. 10b, 11b, and 12b, respectively. It can be seen that larger network size does not affect the performance and behavior of Maxflow and Wide-Short flow.

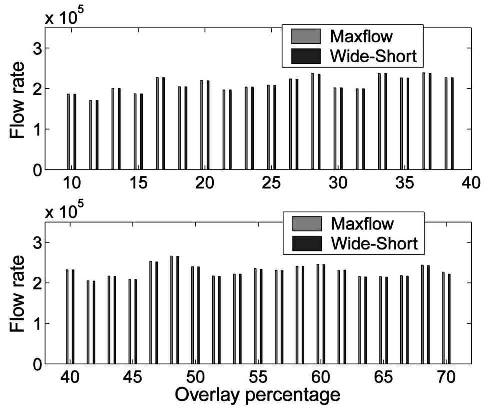
7 Wide-Short Flows in Networks of Various Overlay Ratios

We also experimented with varying the ratio of overlay nodes over a total number of nodes, including low-level nodes in the network. In our simulation, the number of overlay nodes over the total number of nodes in the network ranges between 10 percent and 70 percent in networks of size 300 and 3,000.

Based on the effect of different *DPW* values on Wide-Short flows, we selected the *DPW* values with the aim of obtaining Wide-Short flows that have maximum rates with low delays. We wished to see if Wide-Short flows, with appropriately chosen *DPW*, would be able to attain low delay while still maintaining maximum flow rate, over a wide range of overlay percentages. For network size 300, the *DPW* is set to be 0.05; for network size 3,000, the *DPW* is

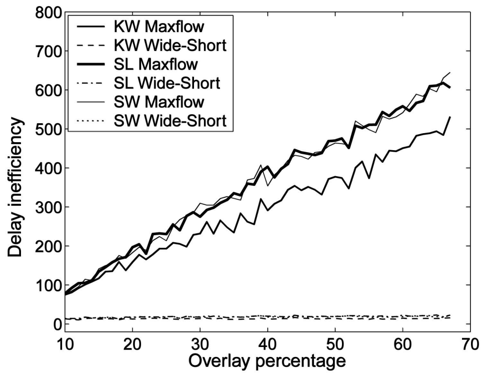


(a)

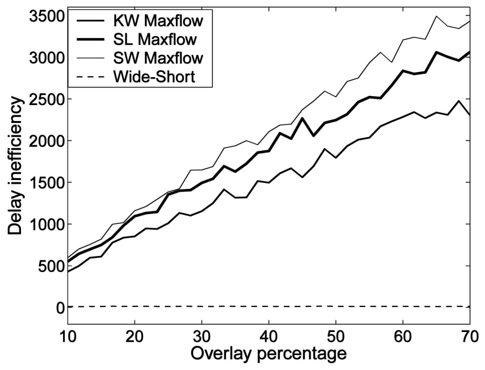


(b)

Fig. 13. Delay inefficiency of Maxflow and Wide-Short flows. In (b), only one Wide-Short curve is shown because the delay results of all three overlay types are too close together to be distinguished. (a) Network size = 300. (b) Network size = 3,000.



(a)



(b)

Fig. 14. (a) Network size = 300. (b) Network size = 3,000.

0.03. As above, three different types of overlays (*KW*, *SL*, and *SW*) are constructed for each network size.

For each network topology and each type of flow (Maxflow, Wide-Short, Shortest Path), 100 pairs of source and destination nodes are randomly selected, the flow rates and delays shown in the following are all averaged over 100 runs.

Fig. 13 shows the average flow rates of Wide-Short flows compared with those of maxflows. The results of only one type of overlay are shown because all three types of overlays yielded the same results of the flow rates of Wide-Short flows matching exactly those of maxflow for all overlay percentages with only the most negligible differences.

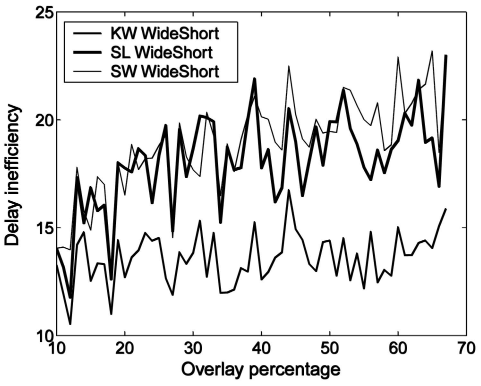
Now, we proceed to examine the delay of Wide-Short flows compared with that of both maxflow and Shortest Path. We define a metric called *delay inefficiency* for comparison: The delay inefficiency of a flow is simply its delay divided by the delay of the Shortest Path in the same overlay topology. It measures how many times worse the delay of this flow is than the smallest possible delay of a flow (one that has only a single path).

The delay inefficiency of Maxflow and Wide-Short flows for the three types of overlays is plotted in Fig. 14. The delay inefficiency of Maxflow in all three types of overlays is so high compared to Wide-Short flows that the latter appears to be 0 when plotted in the same graph with the former. This holds for the entire range of overlay percentages. The delay inefficiency of Maxflow increases linearly, with a large slope, as overlay nodes become more dense.

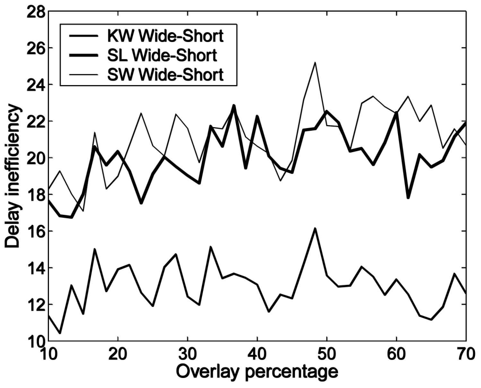
To see more plainly the delay inefficiency of Wide-Short flows, it is shown by itself in Fig. 15. Aside from the obvious fact that Wide-Short flows have a low delay inefficiency that remains well below 30 in all the simulated scenarios, two more things are worth noting based on differences observed between Figs. 14 and 15. 1) For both the smaller and the larger network size, shown in Figs. 15a and 15b, respectively, the delay inefficiency increases only very slightly as overlay percentage increases. 2) As the network size is increased 10-fold (from 300 to 3,000), the delay inefficiency of Wide-Short flows remains roughly the same. In contrast, for Maxflow, delay inefficiency increases by almost an order of magnitude as the network size is increased.

8 Constructing an LCC Overlay

In this section, we present a distributed scheme for constructing an LCC overlay. In Section 2, we showed that node-based LCC exhibits high accuracy and efficiency that are, in most cases, closer to the quality of a complete set of LCC than to that of no LCC. The advantage of node-based LCC is that they are naturally distributed. In our scheme, an overlay node first determines a conservative set of node-based LCC, that is, coarse LCC; it then *successively refines* the LCC.



(a)



(b)

Fig. 15. (a) Network size $= 300$. (b) Network size $= 3,000$.

As with all overlay construction methods, the input is a set of overlay nodes, each possessing a list of other known nodes—the list may not be complete at the beginning, but it is periodically disseminated and updated as new information from other nodes arrive. The existing methods make use of unicast probes to estimate link bandwidth. The independent unicast probes cannot yield information on shared bottlenecks of overlay links. Therefore, the probing tool we use in our scheme is an efficient and accurate technique for detecting shared bottlenecks (DSB), proposed by Katabi et al. in [11] and [12]. This technique is based on the entropy of the interarrival times of packets from flows. A set of flows is partitioned into groups of flows, each group of flows shares a bottleneck, and the bottleneck capacities are also measured. We refer to this probing tool for DSB. Every time DSB is executed with the input of a set of flows, the output is a collection of groups of flows with their corresponding bottleneck capacities.

Prior to determining LCC, a node selects up to k neighbors—any of the existing neighbor selection rules can be substituted here. For our simulation, we used the rule of selecting the k highest bandwidth links.

The node-based LCC are obtained in stages of increasing refinement. In the first stage, the least refined set of LCC is determined. A node executes DSB once with the input of the set of k flows to all its neighbors. The k flows are partitioned into n bottleneck-sharing groups of flows, $g_1, g_2, \ldots, g_n$, with the corresponding bottleneck capacities, $b_1, b_2, \ldots, b_n$. The LCC obtained are thus $C_1 = \{\sum_{e \in g_i} x_e \leq b_i\}_{i=1}^{n}$. Since DSB

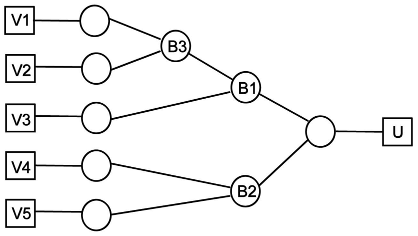


Fig. 16. Illustrates the phenomenon of hidden bottlenecks.

detects only the dominant bottlenecks, some bottlenecks cannot be discovered in the first stage. We give an example of this in Fig. 16; assume that node U is using DSB to probe for bottlenecks, and assume that bottleneck $B1$ has a smaller capacity than $B3$. When node U executes DSB with all five flows from its neighbors $(V1, \ldots, V5)$, only the most dominant bottlenecks $B1$ and $B2$ can be discovered. Now, the set of five flows are partitioned into two groups, $g_1 = \{V1, V2, V3\}$ with bottleneck capacity of $B1$ and $g_2 = \{V4, V5\}$ with capacity of $B2$. The two LCC thus obtained are perfectly accurate but not complete; moreover, they are conservative in bounding the two flows from $V1$ and $V2$ to the capacity of $B1$. To further determine more refined LCC, node U must execute DSB with the input of only the flows from $V1$ and $V2$. This will be done in the second stage of LCC refinement.

In order to guarantee that all hidden bottlenecks behind the dominant ones are found, all possible subsets of flows in each group must be probed separately. However, the brute-force search is exponential in computational complexity and is hence infeasible. We maintain a low complexity by randomly dividing each group g into two subsets and executing DSB on each subset.

Although this procedure does not exhaustively search through all possibilities, our simulation results show that it is not only efficient, but it is also able to find LCC that are negligibly close to the complete LCC after a certain number of refinement stages.

The entire procedure of discovering node-based LCC is summarized as follows:

1. Start with G containing one single group including all k flows.
2. Execute DSB with each group g from G separately.
3. Every group g is partitioned into n subgroups from which n LCC derive; add to C (the growing set of LCC) only those LCC that are not redundant with the ones already in C.
4. Each subgroup containing more than two flows is randomly divided into two subsets, and G now contains all such subsets as its groups.
5. Repeat step 2 as long as more LCC can be found.

We use the simple example in Fig. 16 to illustrate the above procedure for finding node-based LCC. Node U executes the procedure as follows: In step 1, G contains all five flows incident to node U (with the overlay nodes $V1 \ldots V5$). In the first stage, steps 2 finds two LCC (to add to the set C): $f_{V1} + f_{V2} + f_{V3} \leq B1$, $f_{V4} + f_{V5} \leq B2$. Now, there are two subgroups: $\{V1, V2, V3\}$ and $\{V4, V5\}$. The first

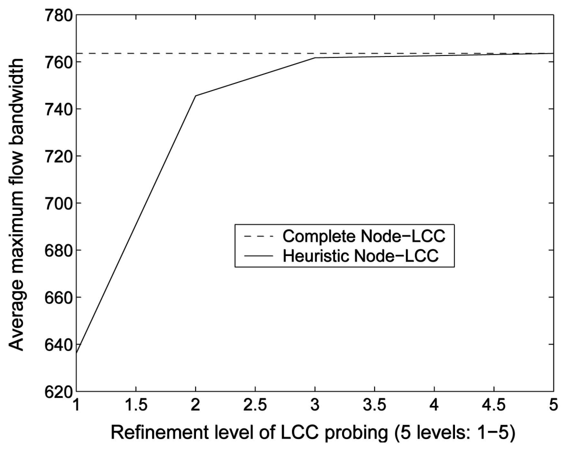


Fig. 17. The rapid convergence of the accuracy of discovered node-based LCC as compared to the complete node-based LCC.

one contains more than two flows and is hence further divided to two subsets: $\{V1, V2\}$ and $\{V3\}$. Thus, at the start of the second stage, G has three groups: $\{V1, V2\}$, $\{V3\}$, and $\{V4, V5\}$. Steps 2 to 4 will then find the new LCC: $f_{V1} + f_{V2} \leq B3$, which will be added to C. At the end, the node U has three node-based LCC in C.

The simulation results for a network of 100 nodes with 30 percent overlay nodes are given in Fig. 17. In our simulation, LCC obtained at successive stages of refinement are used to compute maximum flows and maximum flows are also computed from the complete node-based LCC. A high number of source and destination node pairs are randomly chosen to compute maximum flows and the maximum flow bandwidths are averaged over all such pairs. In Fig. 17, the average maximum flow bandwidths for successive stages of LCC refinement are plotted and compared to the average maximum flow bandwidth computed using complete node-based LCC. After only five refinement stages, the DSB LCC are as good as complete node-based LCC. The number of stages required for such accuracy may have something to do with the node degree limit, which is set at 6 in this simulation, because the node degree limit determines the maximum size of the groups given by DSB.

The complexity of the procedure depends on two factors: the number of executions of DSB and the number of flows probed. The number of packets per flow required for DSB depends on the cross traffic condition and the bottleneck sharing complexity; a reasonable estimate based on reported empirical results in [11] is a few hundred packets. In our simulation, for obtaining LCC that are 98 percent accurate of complete node-based LCC, DSB is executed an average of 3.6 times and the average number of flows probed is 13.5. This translates to a total of a few thousand probes used. It is worth noting, though, that the probing and discovery of node-based LCC can be done passively. The overlay can begin data transmission with an initial topology without knowledge of LCC. Then, gradually through time, the ongoing data transmission essentially acts as passive probing and is used to determine more and more refined node-based LCC. With increasingly complete LCC, the data dissemination topology can be changed accordingly.

9 Related Work

Prior work in overlay networks have without exception assumed an overlay model of independent link capacities with no link correlation. To alleviate the adverse effect of overloading the underlying network from mapping too many overlay flows to shared bottlenecks, the typical approach is to limit overlay node degrees. Our proposed LCC-overlay is able to model the real network topology more accurately by explicitly incorporating link correlation in the succinct form of LCC. We have shown that, through accurate network representation, LCC-overlay ensures a higher overlay quality. To the best of our knowledge, there has not been previous work on the two problems we studied in the context of graphs with LCC: WPC and maximum flow with LCC.

Several projects based on Distributed Hash Tables —CAN [13], Pastry [14], and Chord [15], etc.—designed structured overlay networks in which neighbor mappings are dictated by addresses from abstract coordinate spaces. Distributed algorithms for general-purpose overlay construction were proposed by Young et al. in [16] and by Shen in [4]. The heuristics proposed by Young et al. are to build k interleaved minimum spanning trees. Shen designed and implemented a software layer to build and maintain an overlay using quality metrics of unicast latency and bandwidth and various heuristics. Application-specific proposals have been made for various overlay services, including overlay multicast [1], [17], content distribution [3], [18], and multimedia streaming [19], [2].

Also relevant is the recent work by Ratnasamy et al. [10]. A distributed binning scheme is designed and applied to the formation of unstructured overlay networks. The aim is to incorporate more topological awareness into overlay construction. This work differs from ours in focusing exclusively on the latency metric. The proximity of nodes in latency is the only topological information that is sought. In this work [10], it is assumed that the overlay links can be viewed as essentially independent of each other in terms of latency. In this paper, we focus on accurately aggregating underlying bandwidth information into our overlay model by taking into consideration overlay link capacity correlation.

Common to all these proposals are heuristics that use unicast probing to select overlay routes with low latency or high bandwidth. They view and treat overlay links as independent. However, we propose a new overlay model and, hence, work upon a premise distinct from previous work.

Finally, the well-studied quality-of-service (QoS) literature explores problems that are related to the subject of this paper. Spurred by the increasing demand for multimedia applications over the Internet, the problem of finding routing paths that satisfy certain QoS constraints—that is, the QoS routing problem—has been extensively studied by the research community. Representative examples of such QoS constraints are requirements in bandwidth, delay, jitter, and packet loss. Many QoS routing algorithms have been proposed, for example, [20], [21], [22], [23], and [24]. In such cases, the QoS constraints in question can be thought of as the minimum performance objective (or worst-case performance guarantee) of the algorithm. The problem we consider here is a different one: We want to optimize bandwidth given that the capacity constraints of the overlay links *already exist* in the network as one of its characteristics.

10 CONCLUSIONS

We have introduced a new model of overlay networks, LCC-overlay, that uses LCC to succinctly, efficiently, and accurately represent the real network topology with its link correlations. We have defined the metrics of accuracy and efficiency to qualitatively measure the quality of overlay networks. Through analysis and simulations with realistic network topologies, we showed that LCC-overlay has perfect accuracy and optimal efficiency and even a restricted class of LCC—node-based LCC—yields excellent accuracy and efficiency that are close to complete LCC, whereas overlays with no LCC not only has low quality but its quality deteriorates with increasing network size.

Node-based LCC are more efficient than complete LCC and can be obtained distributedly. We propose a distributed algorithm to construct LCC-overlays with node-based LCC. Simulation showed that the LCC obtained via a probing tool converge rapidly to the complete node-based LCC.

We also studied the problems of WPC and maximum flow with LCC. We proved that WPC is NP-complete, but in realistic overlay topologies, the solutions of regular widest path are almost always the same as the optimal WPC. We formulated maximum flow with LCC as a linear program and proposed an efficient algorithm to solve it. We further studied the problem of optimizing delay while maintaining near-optimal flow rate or bandwidth.

REFERENCES

[1] Y. Chu, S.G. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," *IEEE J. Selected Areas in Comm.*, pp. 1456-1471, Oct. 2002.

[2] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-Bandwidth Multicast in Cooperative Environments," *Proc. 19th ACM Symp. Operating Systems Principles (SOSP '03),* Oct. 2003.

[3] J. Byers and J. Considine, "Informed Content Delivery Across Adaptive Overlay Networks," *Proc. ACM SIGCOMM,* Aug. 2002.

[4] K. Shen, "Structure Management for Scalable Overlay Service Construction," *Proc. Symp. Networked Systems Design and Implementation (NSDI '04),* 2004.

[5] A. Medina, A. Lakhina, I. Matta, and J. Byers, *BRITE: Boston Univ. Representative Internet Topology Generator,* http://www.cs.bu.edu/brite, 2007.

[6] C. Faloutsos, M. Faloutsos, and P. Faloutsos, "On Power-Law Relationships of the Internet Topology," *Proc. ACM SIGCOMM,* Aug. 1999.

[7] H. Tangmunarunkit et al., "Network Topology Generators: Degree-Based versus Structural," *Proc. ACM SIGCOMM,* 2002.

[8] M.S. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.

[9] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[10] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-Aware Overlay Construction and Server Selection," *Proc. IEEE INFOCOM,* 2002.

[11] D. Katabi and C. Blake, "Inferring Congestion Sharing and Path Characteristics from Packet Interarrival Times," technical report, Laboratory of Computer Science, Massachusetts Inst. of Technology, 2001.

[12] D. Katabi, I. Bazzi, and X. Yang, "A Passive Approach for Detecting Shared Bottlenecks," *Proc. IEEE Int'l Conf. Computer Comm. and Networks (ICCCN '01),* 2001.

[13] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM,* pp. 149-160, Aug. 2001.

[14] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. IFIP/ACM Middleware,* Nov. 2001.

[15] I. Stoica, R. Morris, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM,* 2001.

[16] A. Young, J. Chen, Z. Ma, A. Krishnamurthy, L. Peterson, and R. Wang, "Overlay Mesh Construction Using Interleaved Spanning Trees," *Proc. IEEE INFOCOM,* 2004.

[17] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," *Proc. ACM SIGCOMM,* Aug. 2002.

[18] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," *Proc. ACM Symp. Operating System Principles (SOSP '03),* 2003.

[19] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking," *Proc. 12th Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '02),* May 2002.

[20] J.M. Jaffe, "Algorithms for Finding Paths with Multiple Constraints," *Networks,* pp. 95-116, 1984.

[21] A. Iwata et al., "ATM Routing Algorithms with Multiple QoS Requirements for Multimedia Internetworking," *IEICE Trans. and Comm. E79-B,* pp. 999-1006, 1996.

[22] H. De Neve and P. Van Mieghem, "TAMCRA: A Tunable Accuracy Multiple Constraints Routing Algorithm," *Computer Comm.,* pp. 667-679, 2000.

[23] P. Van Mieghem, H. De Neve, and F.A. Kuipers, "Hop-by-Hop Quality of Service Routing," *Computer Networks,* pp. 407-423, Oct. 2001.

[24] F.A. Kuipers and P. Van Mieghem, "MAMCRA: A Constrained-Based Multicast Routing Algorithm," *Computer Comm.,* pp. 801-810, May 2002.

Ying Zhu received the BSc degree in computer science and mathematics from Dalhousie University, the MSc degree in numerical analysis also from the University of Toronto, and the PhD degree in computer engineering from the University of Toronto, Toronto. He is an assistant professor in the faculty of Business and Information Technology, University of Ontario Institute of Technology. Her main research interests include overlay networks over heterogeneous substrates (wired and wireless networks), peer-to-peer networks, and network optimization.

Baochun Li received the MS and PhD degrees in computer science from the University of Illinois, Urbana-Champaign, and the BE degree in computer science from Tsinghua University, Beijing. He is an associate professor in the Department of Electrical and Computer Engineering, University of Toronto. His main research interests are in application-layer algorithms in a wide variety of distributed systems, including peer-to-peer and overlay networks, multihop wireless networks, and sensor networks. He is a senior member of the IEEE.