Overlay Multicast with Inferred Link Capacity Correlations

Ying Zhu, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto
{*yz, bli*}*@eecg.toronto.edu*

Abstract

We model the overlay using linear capacity constraints, which accurately and succinctly capture overlay link correlations. We show that finding a maximum-bandwidth multicast tree in an overlay with LCC is NP-complete and propose an efficient distributed heuristics algorithm.

1 Introduction

Overlay networks are distinct from regular networks in having a two-level structure. The high-level consists of overlay nodes (end-systems) and links (unicast connections); the low-level is a densely connected IP network of routers and physical links. We say overlay links are *correlated* when they map to paths that overlap: the *sum* of the capacities of the overlay links is constrained by the capacity of the shared physical link.

Some existing work, however, views the overlay as a regular network graph where capacities of links are independent and scalars (unicast bandwidths). We refer to this as the independent overlay model. Other related work, *e.g.*, [10, 8], either focus on end-to-end latencies, or aims to discover the underlying IP-layer topology. However, it is infeasible to discover the entire AS-level topology using probing techniques (such as `traceroute`).

We propose *linear capacity constraints* (LCC) as a formulation of overlay link correlations. The problem we study is finding maximum-bandwidth overlay multicast trees. We define three metrics to measure the quality of these trees: accuracy, efficiency and stress. We show that the problem of finding a maximum-bandwidth multicast tree in an LCC-overlay is NP-complete, and propose a heuristics algorithm to solve it. Two types of LCC are considered: Complete-LCC and Node-LCC. Our algorithm always obtains near-optimal trees, not only with complete LCC, but even with the restricted and inherently distributed node-based LCC. Simulation results show that the algorithm converges quickly and is scalable for increasing network sizes. Furthermore, we discuss distributed construction of an LCC-overlay with node-based LCC by employing certain probing techniques, and propose a distributed variation of our algorithm.

Remark. To ensure high-bandwidth overlay multicast, the overlay network must have knowledge of the underlying network topology. Rather than discovering the entire underlying topology, the formulation of LCC is a *succinct and accurate abstraction* of it; LCC provide *sufficient* information to capture overlay link correlations. A surprising and encouraging result is that even with the incomplete and distributed Node-LCC, near-optimal bandwidth multicast can be obtained.

The remainder of the paper is organized as follows. We summarize related work and distinguish our work in Sec. 2. In Sec. 3, we present LCC and the metrics. The problem of finding multicast trees is studied in Sec. 4, as well as the heuristics algorithm. We evaluate the algorithm in Sec. 5, discuss its distributed variation in Sec. 6, and conclude in Sec. 7.

2 Related Work

To the best of our knowledge, there has not been previous work on the problem of highest-bandwidth multicast tree with LCC that we studied in this paper. Overlay networks and multicast have been extensively studied in [12, 11, 13, 3, 1, 7, 2]. Common to all these proposals are that they view and treat overlay links as independent. Recent work by Ratnasamy *et al.* [10] aims to incorporate more topological awareness into overlay construction. This work differs from ours in focusing exclusively on latency. We have studied the new problem of maximum-bandwidth multicast tree in LCC-overlays; our previous paper [15] did not study multicast. Definitions of the metrics are also new.

In [6], Kim *et al.* propose a protocol to eliminate probed bottlenecks in an overlay multicast tree. Our work is distinct from theirs in formally abstracting link correlations using linear capacity constraints. We also rigorously define the metrics of accuracy, efficiency and stress, which provide both quantitative performance measurement and crucial insight.

3 Linear Capacity Constraints

In the independent model, an overlay network is viewed as a weighted graph in which overlay links have scalar uncorrelated capacities. A generic scheme of constructing an overlay with the aim of optimizing bandwidth is: each node selects its neighbors by choosing k adjacent links with the highest unicast bandwidth. The imposition of a node degree limit k is a commonly used heuristic to alleviate overloading in the low-level network. Since this is representative of overlay construction schemes from previous work, we adopt it to assess the independent model.

We use the example in Figure 1 to illustrate the disadvantage of the independent model, and to introduce and show the advantage of the LCC model.

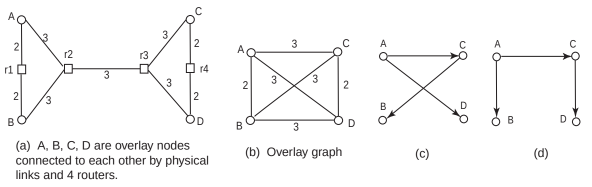

A
B
C
D
r1
r2
r3
r4
2
3
(a) A, B, C, D are overlay nodes connected to each other by physical links and 4 routers.
(b) Overlay graph
(c)
(d)

Figure 1. A simple example illustrating the disadvantage of the independent model and the advantage of linear capacity constraints.

Consider the low-level network in Figure 1(a). The overlay graph is given in Figure 1(b), in which the overlay links are labeled by their independent unicast bandwidths. Figure 1(b) is the case when k is 3 in the above construction scheme based on the independent model; it is not hard to see that the following reasoning will also hold for $k = 2$ or 1.

In the independent model, the highest-bandwidth multicast tree in this overlay graph[1] is shown in Figure 1(c). Although the predicted bandwidth of the tree is 3, it can be seen that the achievable bandwidth of the tree is 1 because all three links in the tree share a single bottleneck physical link $(r2, r3)$ with capacity 3.

In the LCC model, the capacity of each overlay link is represented by a *variable* and a set of linear constraints are used to formulate link correlations. The complete set of linear capacity constraints (LCC) for the example is given below in matrix form:

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

[1]The tree can be found by an all-widest paths algorithm that is a variant of Dijkstra's all-shortest paths.

Using these LCC, the highest-bandwidth tree obtained is given in Figure 1(d). In this case, the achievable bandwidth is 2, the same as the predicted bandwidth. This corresponds exactly to the actual optimal tree for this network.

From the above example, we can observe that the achievable bandwidth of the independent overlay is low because it does not incorporate link correlations and therefore has insufficient information to obtain a high-bandwidth tree. In the LCC-overlay, however, the formulation of link correlations as LCC results in an accurate abstraction of underlying topology, hence an optimal multicast tree can be found.

3.1 Formal definitions of LCC and metrics

We now present the definitions formally. The two-level hierarchy of an overlay network can be formulated as consisting of: (1) A low-level (IP) graph $G = (V, E)$; each low-level link $e \in E$ has a capacity of $c(e) \geq 0$. (2) A high-level overlay graph $\widehat{G} = (\widehat{V}, \widehat{E})$, where $\widehat{V} \subset V$; (3) 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 independent overlay is defined as follows.

Definition 1 (Independent overlay): The *independent 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 2 (LCC-overlay): The *LCC-overlay* is a triplet $(\widehat{G}, C, b)$ where (1) The capacity of each link $\widehat{e}$ in $\widehat{G}$ is a variable $x_{\widehat{e}}$. (2) (C, b) represent a set of m linear capacity constraints $Cx \leq b$: 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; $b \in \mathrm{R}^m$ is the capacity vector.

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

3.1.1 Overlay (or predicted) bandwidth

Given any tree T in either the independent overlay or the LCC-overlay, we observe that there exists the notion of the *overlay bandwidth* of T, or its *predicted bandwidth* in the overlay model of interest. Because algorithms rely solely on predicted bandwidth to optimize tree formation, this is an crucial concept.

For a tree T in the independent overlay $(\widehat{G}, \widehat{c})$, the overlay bandwidth of T is the minimum bandwidth link in T as given by $\widehat{c}$, or formally, $\min\{\widehat{c}(\widehat{e}) : \widehat{e} \in T\}$.

As for a tree T in an LCC-overlay $(\widehat{G}, C, b)$, the overlay bandwidth of T can be obtained by the following procedure. For each constraint (C_i, b_i) (i-th row in C, b), compute the bandwidth allocated to every link $j \in T$ for which $C_{i,j} = 1$: $\gamma_i(j) = b_i / \sum_{k \in T} C_{i,k}$. That is, $\gamma_i(j)$ is the bandwidth allocated to j in T by the i-th constraint. Considering all constraints together, the allocated bandwidth of link j in T is $\gamma j = \min\{\gamma_i(j) : i = 1 \ldots m\}$, where m is number of rows in C. Finally, the overlay bandwidth of T is $\sigma(T) = \min\{\gamma(j) : j \in T\}$.

3.1.2 Achievable bandwidth $\sigma_G(T)$

Another notion of interest is: Given any overlay tree T, what is the *achievable bandwidth* of T? Let $\sigma_G(T)$ denote the achievable bandwidth of T in overlay $\widehat{G}$ residing on top of G. The procedure for obtaining $\sigma_G(T)$ is summarized in Figure 2. The following verifies its correctness.

Proposition: Given an overlay $\langle \widehat{G}, G, P \rangle$ and any overlay tree $T \subseteq \widehat{G}$, the highest bandwidth that T can achieve is $\sigma_G(T)$ obtained by the procedure in Fig. 2.

Proof: Due to space constraint, please refer to [14] for the proof.

```
for each $e \in E$
   Allocate $c(e)$ equally among
      $\{\widehat{e} : e \in P(\widehat{e})$ and $T(\widehat{e}) > 0\}$,
   let each allocation be denoted by $\gamma_T^e(\widehat{e})$
   for each $\widehat{e} \in \widehat{E}$
      if $T(\widehat{e}) > 0$     $\gamma_T(\widehat{e}) \leftarrow \min\{\gamma_T^e(\widehat{e}) : e \in P(\widehat{e})\}$
      else             $\gamma_T(\widehat{e}) \leftarrow 0$
   $\sigma_G(T) \leftarrow \min\{\gamma_T(\widehat{e}) : \gamma_T(\widehat{e}) \neq 0$ and $\widehat{e} \in \widehat{E}\}$
```

Figure 2. The procedure for obtaining achievable bandwidth of a tree T in G, $\sigma_G(T)$.

We proceed now to present (original) definitions of the metrics: accuracy, efficiency and stress. Accuracy is the ratio of the overlay or predicted bandwidth of a tree over its achievable bandwidth. Efficiency is the ratio of achievable bandwidth of a tree over the optimum tree bandwidth. Stress is defined for low-level links that are mapped by overlay tree links; it is essentially the load placed on a low-level link by the overlay tree, normalized by the link capacity. The formal definitions are given below.

Definition 3 (Accuracy): The *accuracy* of a multicast tree T in an overlay network $\langle G, \widehat{G}, P \rangle$, is (overlay bandwidth of T in $\widehat{G}$)$/(\sigma_G(T))$.

Definition 4 (Efficiency): The *efficiency* of a multicast tree T in an overlay network $\langle G, \widehat{G}, P \rangle$, is $(\sigma_G(T))/(\sigma_G(T_{\text{opt}}))$, where T_{opt} is an optimal multicast tree in $\langle G, \widehat{G}, P \rangle$.

Definition 5 (Stress): The *stress* of a low-level edge $e \in E$ due to a multicast tree T in an overlay network $\langle G, \widehat{G}, P \rangle$, is $|\{\widehat{e} : \widehat{e} \in \widehat{E}$ and $\widehat{e} \in T\}|/c(e)$, where $c(e)$ is the capacity of e.

4 Multicast Tree with Linear Capacity Constraints

4.1 MTC is NP-complete

We showed above that by using LCC, the bandwidth of multicast trees can be significantly increased. Now we are faced with the question of how to obtain the highest-bandwidth multicast tree in an LCC-graph.

We state the problem of highest-bandwidth multicast tree with LCC (MTC) as a decision problem as follows.

INSTANCE: An LCC-graph (G, C, b), where $G = (V, E)$ and (C, b) are LCC; a positive integer B.

QUESTION: Is there a multicast tree T for G such that the bandwidth of T is $\geq B$?

Theorem: MTC is NP-complete.

Proof: Due to space constraints, please refer to [14] for the proof.

4.2 Heuristics Algorithm

We propose a heuristics algorithm to solve the problem. The input is an LCC-graph $(\widehat{G}, C, b)$. The goal is to find a high-bandwidth multicast tree. The main idea of the algorithm is to begin with an initial tree, and make incremental improvements by replacing, at each iteration, the lowest-bandwidth edge with a higher-bandwidth one, thereby increasing bandwidth of the tree. For every edge replacement, the algorithm preserves the topology of a multicast tree, *i.e.*, a tree spanning all receiver nodes.

The initial multicast tree is found by first forming a regular graph G with edges weighted with independent edge capacities (numbers). This is easily done by setting $\min\{b_i : C(i, e) = 1\}$ to be the capacity for each edge $e \in \widehat{G}$. Note that this corresponds exactly to an independent overlay with unicast bandwidths for the edges. A highest-bandwidth multicast tree is found in G and set to be the initial tree, let it be denoted T_0. The algorithm then incrementally improves the tree by always replacing the worst edge with a better edge, if possible, while maintaining the spanning tree structure.

The algorithm, which we call *HMTC* (High-bandwidth Multicast Tree with LCC), is summarized in Table 4.

We re-use the example overlay network in Figure 1 from Sec. 3 to illustrate the algorithm. The overlay graph along with the LCC (C, b) are in Figure 3(a). For simplicity of presentation, we let UV denote the capacity variable of edge (U, V).

In the graph in Figure 3(a), the numbers labeling the edges are the independent edge capacities. For the graph with independent edge capacities, a highest-bandwidth multicast tree is found: T_0 in Figure 3(b). The *LCC of* T_0 is listed on the right of T_0 in the figure. The LCC of T_0, C_{T_0}, is LCC (of the graph) (C, b) intersected with T_0 so that they only contain capacity variables of edges from T_0.

In this instance, there is only one constraint in LCC of T_0: $AC + AD + BC \leq 3$. Each T_0 edge is thus allocated an equal bandwidth of 1. The bandwidth of T_0 is the minimum of allocated bandwidths of its edges, in this case, 1.

The worst edge, one with the lowest bandwidth allocated by the LCC of T_0, is selected to be replaced. Since all three edges have lowest bandwidth, any one can be selected, say edge AC. We want to find another edge XY such that by removing AC and adding XY to form T_1, T_1 is a multicast tree and the bandwidth of T_1 (allocated by the LCC of T_1) is higher than the bandwidth of T_0. In other words, XY connects the subtree under AC, *i.e.*, subtree rooted at node C, with the rest of the tree. The resulting new tree also has improved bandwidth.

Edge CD satisfies the above conditions; the removal of AC and the new tree T_1 formed by adding CD can be seen in Figure 3(c) and (d), respectively. The LCC of T_1 and the bandwidths allocated to the edges are given as well. The bandwidth of T_1 is 1.5, an improvement on T_0.

Now the above edge replacement procedure is repeated for T_1. The new tree T_2 is shown in Figure 3(f); its bandwidth is 2. At this point, none of the three edges can be replaced by another edge to improve the tree bandwidth and the algorithm terminates.

```
HMTC($\widehat{G}, C, b$)
 1  obtain $G$ with independent edge
      capacities from $\widehat{G}$ and LCC $(C, b)$
 2  $T_0 \leftarrow$ highest-bandwidth tree in $G$
 3  $(C_{T_0}, b) \leftarrow$ LCC of $T_0$
 4  $R \leftarrow$ edges in $T_0$ in ascending order
      of bandwidths allocated by $(C_{T_0}, b)$
 5  $T_{hi} \leftarrow T_0$
 6  iter $\leftarrow 1$
 7  while iter $\leq$ max_num_iterations and $R \neq \varnothing$
 8    $r \leftarrow$ 1st edge in $R$
 9    $R \leftarrow R - \{r\}$
10    $T_{sub} \leftarrow$ subtree under $r$
11    $T \leftarrow T_{hi} - \{r\}$
12    for each edge $e \in T - T_{sub}$
13      $T' \leftarrow T \cup \{e\}$
14      $C_{T'} \leftarrow$ LCC of $T'$
15      if bandwidth$(T')$ > bandwidth$(T_{hi})$
16        $T_{hi} \leftarrow T'$
17        $R \leftarrow$ edges in $T_{hi}$ in ascending
18          order of allocated bandwidths
19        break out of for-loop
20    iter $\leftarrow$ iter $+ 1$
21  return $T_{hi}$
```

Figure 4. Summary of heuristics algorithm *HMTC*.

4.3 Effectiveness of HMTC algorithm in minimizing stress

We note that to maximize multicast bandwidth, one should minimize the maximum link stress in the underlying network. The *HMTC* algorithm attempts to accomplish exactly this. To analyze this, we *visually* examine low-level link stress. For ease of visual representation, we find it convenient to define *stress ratio*: for low-level link e, it is the ratio of stress of e over the minimum stress of all low-level links with at least one overlay link mapped to them, rounded to an integer. For instance, a stress ratio of 3 means the links has three times the stress of a link with stress ratio 1. We will refer to stress ratio as stress, since the former is a certain normalized form of the latter. A *stress graph* is a graph of low-level links that have overlay links mapped to them, and for each link between a pair of nodes (u, v), the number of edges connecting u and v in the stress graph is the stress of link (u, v).

Just to illustrate how *HMTC* incrementally minimizes the maximum link stress after each iteration of the algorithm, we revisit our example in Fig. 3 from Sec. 4.2. In Fig. 5, we show the stress graphs of the trees constructed at each iteration of *HMTC*. It can be seen that the worst link stress progressively decreases as the algorithm progresses.

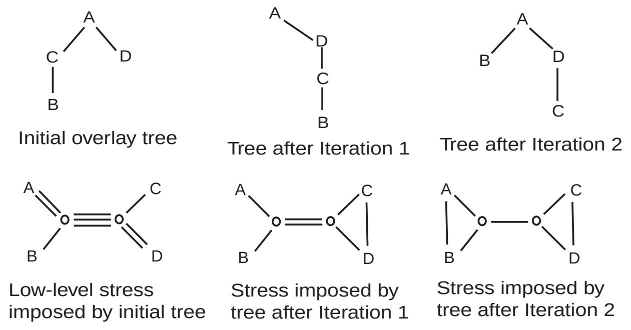

A
C
D
B
Initial overlay tree
A
D
C
B
Tree after Iteration 1
A
B
D
C
Tree after Iteration 2
A
C
B
D
Low-level stress imposed by initial tree
A
C
B
D
Stress imposed by tree after Iteration 1
A
C
B
D
Stress imposed by tree after Iteration 2

Figure 5. Incremental improvement in stress throughout execution of *HMTC* on the example network. The top four graphs are overlay and the bottom four are corresponding underlying network.

5 Evaluation of Algorithm Performance and Convergence

In this section, we evaluate the performance and convergence of our heuristics algorithm *HMTC*. We consider complete LCC — the complete set of LCC that expresses all link correlations, and node-based LCC — every constraint contains only links incident to the same node. We compare three cases: optimal independent tree (*Independent*), *HMTC* with complete-LCC (*Complete-LCC*) and *HMTC* with node-LCC (*Node-LCC*). To generate realistic network topologies, we use a power-law Internet topology generator, BRITE [9].

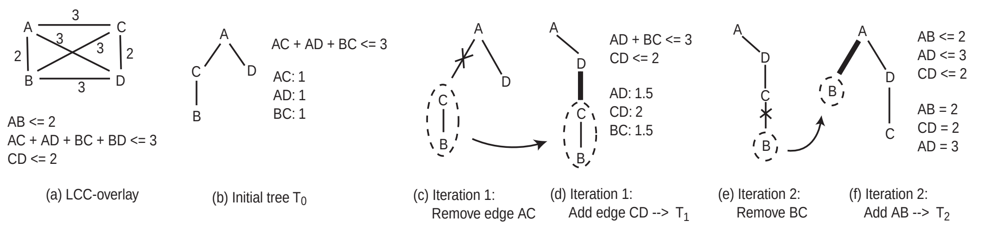


Figure 3. An example of the heuristics algorithm being executed.

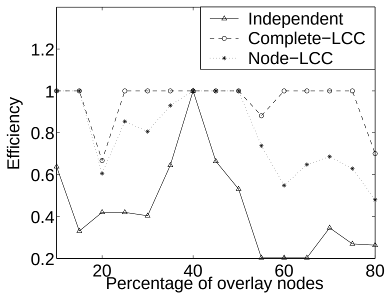


Figure 6. Efficiency versus overlay ratio, low-level size = 300.

5.1 Performance metrics

The performance metrics we use are efficiency, accuracy and stress, as defined previously in Sec. 3.1. For stress, we continue to use stress ratio (and stress graph) for evaluation purposes. We will use the terms 'stress' and 'stress ratio' interchangeably. Recall that the *efficiency* of a tree T measures how close the bandwidth of T is to the optimal, specifically, it is the ratio of achievable bandwidth of T over optimal multicast tree bandwidth. However, as we showed above, finding the optimal in LCC-overlays is NP-complete. Therefore, we use an *upper bound* of the optimal: the optimal *independent* tree bandwidth.

5.2 Efficiency and Accuracy

First, we study the effect of different ratios of overlay size to low-level size, by fixing low-level network size to 300 and varying the percentage of overlay nodes from 10% to 80%. Efficiency for the three types of trees is plotted against the ratio of overlay to low-level size, in Figure 6. The efficiency of Complete-LCC is nearly always 1 or optimal. Node-LCC follows Complete-LCC remarkably closely in efficiency for all overlay-to-network ratios of less than 60%, i.e., all realistic overlay percentages. Efficiency of Independent trees is by far the lowest. The shape of the curve can be explained. When overlay nodes are few compared to the low-level network size, the paths between them are likely long and contain more links, resulting in higher probabilities of the paths intersecting and sharing links. The likelier overlay link correlations are, the higher the probability that some shared low-level link is forced to allocate equally low bandwidth to overlay links. Things are not necessarily rosier with high densities of overlay nodes. Higher percentage of overlay nodes leads to more overlay links mapped to the same low-level network, naturally, collisions increase. Intuitively, there might exist a middle ground where paths are not so long and overlay links are not so dense, such that overloading of bottleneck links is minimized, thereby achieving the optimal. This optimum point seems to be at 40% here.

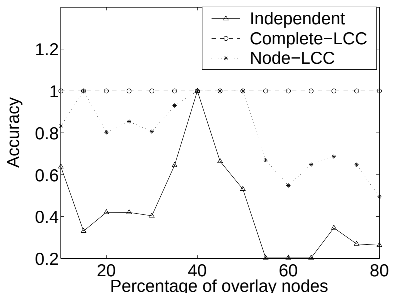


Figure 7. Accuracy versus overlay ratio, low-level size = 300.

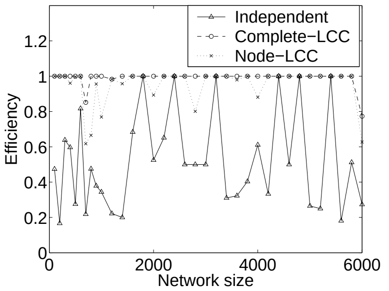


Figure 8. Efficiency versus network size, 10% overlay nodes.

The plot of accuracy versus overlay percentage can be seen in Figure 7. It is obvious that Complete-LCC is always

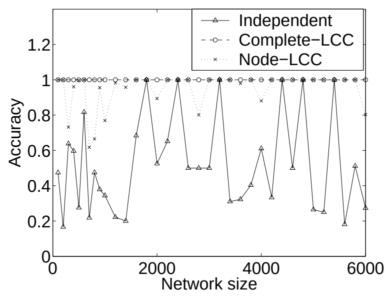


Figure 9. Accuracy versus network size, 10% overlay nodes.

perfectly accurate (*i.e.*, accuracy of 1), due to its complete link correlation information. An encouraging observation here is that Node-LCC accuracy is greater than 80% for all realistic overlays ($< 60\%$). Note that for Independent trees, accuracy is exactly the same as efficiency, the modified version. The accuracy curves are in general extremely similar to the efficiency, which is evidence that supports our proposition that to ensure high multicast bandwidth, it is *necessary* for an overlay to *accurately* represent the true network topology.

The next parameter we vary is the total network size, from 100 to 6000 nodes, while keeping the percentage of overlay nodes at a constant 10%, a value which we believe is realistic. Efficiency is shown in Figure 8 and accuracy in Figure 9. In this experiment, the efficiency and accuracy graphs are almost indistinguishable. Again, Node-LCC performs almost as well as the best Complete-LCC, with Independent being much worse, for almost all network sizes.

5.3 Stress

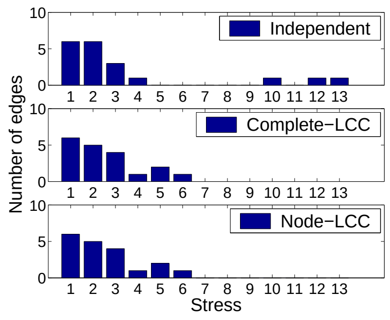


Figure 11. Distributions of link stress for Independent, Complete-LCC and Node-LCC, network size = 100, 10% overlay nodes.

To further investigate the underlying cause of poor and good performance, we visually examine low-level link stress. With a fixed network size and overlay percentage, stress graphs for each of the three trees (Independent, Complete- and Node-LCC) are shown. The stress graph contains only low-level edges with at least one overlay tree link mapped to it. For every low-level edge (u, v) with stress s, s multiple edges are drawn between u and v.

Consider the scenario of network size 100 and overlay ratio 10%; it corresponds to the 100 point on the x-axis of the efficiency graph in Figure 8. The stress graphs are given in Fig. 10. The Independent stress graph, in Figure 10(a), clearly shows three heavily stressed links. This is confirmed by the distribution of stress shown in the top histogram in Figure 11: three links have higher stress. The stress graphs of Complete-LCC and Node-LCC, in Figure 10(b) and (c), look the same. Their distributions of stress, bottom two histograms in Figure. 11, show that the maximum stress is half of that for Independent.

Herein lies the reason for optimal efficiency of Complete-LCC and Node-LCC and for poor efficiency of Independent, as seen in Figure 8 at the x-value of 100. The maximum link stress placed by the Independent tree is twice as high as the maximum link stress caused by the two LCC trees. Independent efficiency being roughly half of that of the two LCC is a convincing indication that our definition of stress is reasonable and useful.

We examine another scenario where network size is 700 with 10% overlay nodes. The stress graphs in Figure 12 show Node-LCC and Complete-LCC to be similar, while stress for Independent is clearly higher in several links. Figure 13 informs that Complete-LCC is only slightly better than Node-LCC, and Independent has some links with much worse stress. This is reflected in their respective efficiencies, in the corresponding 7th point on the efficiency curves in Figure 8.

We observe that the virtue of *HMTC* with LCC is its success in choosing overlay links that spread load evenly among the low-level links, minimizing stress placed on any single link. The success is prominent even with the much restricted Node-LCC.

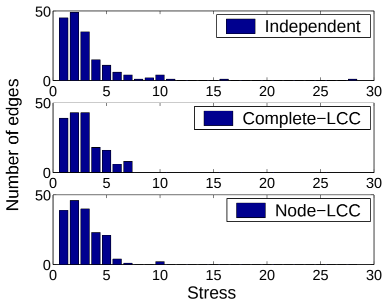
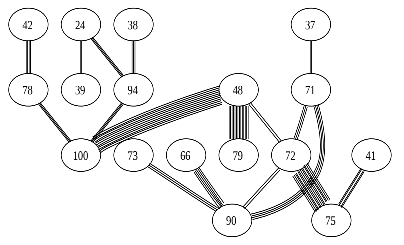
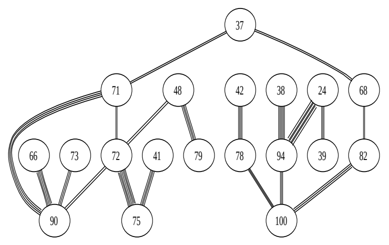


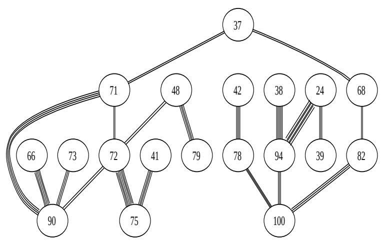
Figure 13. Distributions of link stress for Independent, Complete-LCC and Node-LCC, network size = 700, 10% overlay nodes.



(a) Stress graph for Independent, network size = 100.

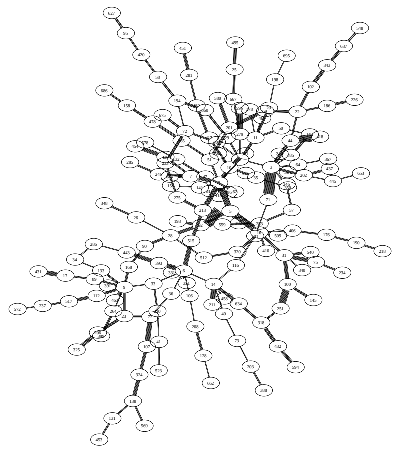


(b) Stress graph for Complete-LCC, network size = 100.

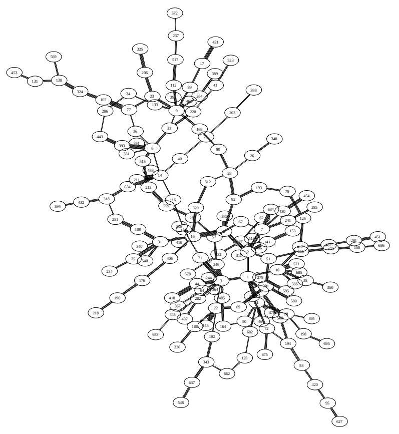


(c) Stress graph for Node-LCC, network size = 100.

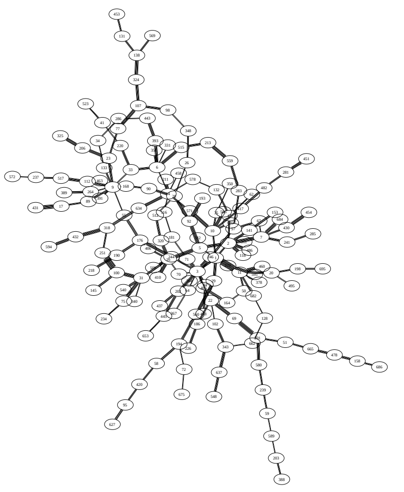
Figure 10. Stress graphs, network size = 100.



(a) Stress graph for Independent, network size = 700.



(b) Stress graph for Complete-LCC, network size = 700.



(c) Stress graph for Node-LCC, network size = 700.

Figure 12. Stress graphs, network size = 700.

5.4 Convergence and Scalability

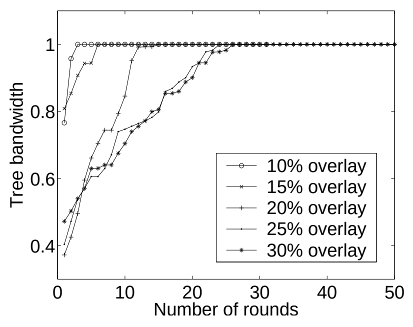


Figure 14. Convergence of heuristics algorithm for various overlay ratios, low-level size = 300.

We analyze the convergence of our algorithm *HMTC* in Figure 14. For a fixed network of 300 nodes and increasing overlay ratios, the (normalized) tree bandwidth attained is plotted against the number of rounds that has been executed in the algorithm. The final bandwidth is always reached in a small number of rounds, for all overlay ratios. Although the number of rounds required to converge increases slightly for higher overlay ratios, it remains small. We do not show them for clarity, but for all ratios up to 80%, the algorithm converges in less than 50 rounds. Scalability of the algorithm is supported by the fact that similar convergence occurs for network sizes up to the high thousands.

6 Distributed Multicast in LCC-Overlay

Above simulation results showed that *HMTC* can find near-optimal trees with the inherently distributed Node-LCC. Two obstacles remain: (1) How do we practically construct an LCC-overlay in a decentralized fashion? (2) The algorithm *HMTC* must be modified to become fully distributed.

We have previously proposed, in [15], a decentralized scheme for constructing an LCC-overlay. It is based on an existing efficient technique for detecting shared bottlenecks, proposed by Katabi et al. in [5, 4]. The node-based LCC

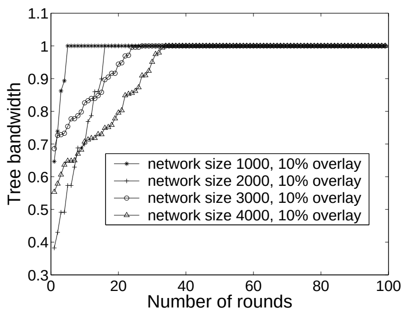


Figure 15. Convergence of heuristics algorithm for various network sizes with 10% overlay nodes.

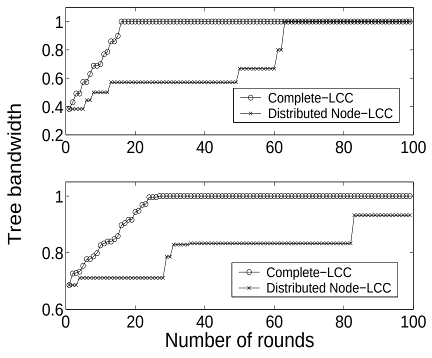


Figure 16. Top: Convergence for network size 1000. Bottom: Convergence for size 4000. (10% overlay)

are obtained in iterations of increasing refinement, which showed fast convergence in simulations. Due to space constraints, we do not go into the details.

Once the LCC-overlay is constructed and maintained, one needs only to develop a distributed version of *HMTC* using node-based LCC. The distributed algorithm differs from the centralized version in two aspects. In the centralized algorithm, global link replacement is done — the worst link is replaced by another that improves the tree bandwidth. In the distributed algorithm, every node does local replacement of its worst incident link in the tree, by one of its incident links in the overlay; the replacement is done whenever the new link has a higher bandwidth allocated by the LCC (not only when bandwidth of the entire tree is increased).

The other difference is the new mechanism for cycle avoidance in the distributed algorithm. When nodes make local link replacements, cycles may be introduced and it would no longer be a tree. In order to preserve the tree structure, whenever a node u is beginning the process of replacing a local link, a ”replacing” notification is passed down to all nodes in the subtree rooted at u; when u is finished, a ”finished” notification is passed down. When a node v receives a ”replacing” notification from its parent, v forwards it to its children, and *does not accept any requests to add new links incident to it* by other nodes who may be doing their own local adjustments — until v receives the ”finished” notification.

Simulation results in Figure 16 show that the distributed algorithm based on node-based LCC converges to the optimal or near-optimal, in less than 100 rounds for relatively large networks.

7 Conclusions

In this paper, we have studied the problem of high-bandwidth overlay multicast in overlay networks with linear capacity constraints. We showed that it is NP-complete and proposed a heuristics algorithm *HMTC*. Extensive simulation results demonstrated that *HMTC* achieves near-optimal performance even with Node-LCC, for which we propose a distributed scheme of constructing.

References

[1] J. Byers and J. Considine. Informed Content Delivery Across Adaptive Overlay Networks. In *Proc. of ACM SIGCOMM,* August 2002.

[2] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-Bandwidth Multicast in Cooperative Environments. In *Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003),* October 2003.

[3] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. A Case for End System Multicast. *IEEE Journal on Selected Areas in Communications*, pages 1456–1471, October 2002.

[4] D. Katabi, I. Bazzi, and X. Yang. A passive approach for detecting shared bottlenecks. In *Proc. of ICCCN '01,* 2001.

[5] D. Katabi and C. Blake. Inferring Congestion Sharing and Path Characteristics from Packet Interarrival Times. Technical report, Laboratory of Computer Science, Massachusetts Institute of Technology, 2001.

[6] M. Kim, Y. Li, and S. Lam. Eliminating Bottlenecks in Overlay Multicast. In *Networking 2005*, 2005.

[7] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *Proc. of ACM SOSP*, 2003.

[8] M. Kwon and S. Fahmy. Path-aware Overlay Multicast. *Computer Networks*, 47(1):23–45, January 2005.

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

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

[11] K. Shen. Structure Management for Scalable Overlay Service Construction. In *Proc. of NSDI*, 2004.

[12] I. Stoica, R. Morris, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of ACM SIGCOMM*, 2001.

[13] 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.

[14] Y. Zhu and B. Li. Overlay Multicast with Inferred Link Capacity Constraints. Technical report, Department of ECE, University of Toronto, 2001.

[15] Y. Zhu and B. Li. Overlay Networks with Linear Capacity Constraints. In *to appear in the Proceedings of the Thirteenth IEEE International Workshop on Quality of Service (IWQoS 2005)*, 2005.