The performance of locality-aware topologies for peer-to-peer live streaming

Richard G. Clegg [*], David Griffin[†], Raul Landa[‡], Eleni Mykoniati[§] and Miguel Rio[¶][||]

Abstract

This paper is concerned with the effect of overlay network topology on the performance of live streaming peer-to-peer systems. The paper focuses on the evaluation of topologies which are aware of the delays experienced between different peers on the network. Metrics are defined which assess the topologies in terms of delay, bandwidth usage and resilience to peer drop-out. Several topology creation algorithms are tested and the metrics are measured in a simple simulation testbed. This gives an assessment of the type of gains which might be expected from locality awareness in peer-to-peer networks.

1 Introduction

This paper investigates the impact of the topology of overlay networks on performance metrics for peer-to-peer live streaming. An overlay network is a conceptual network of peers which exists on top of the standard Internet. Peers on the overlay network connect according to given rules to form a topology. There has been recent research interest in making overlay networks locality-aware for so that peers may more easily find "nearby" peers. In this paper we undertake a systematic evaluation of a number of alternative locality-aware topology construction methods (and some random methods for comparison).

The situation considered is that of a single node, known as the *peercaster* wishing to distribute live streaming content through a peer-to-peer network. The peers in the network wish to download this content reliably and with a low delay between the peercaster and themselves. The challenge of distributing live content is somewhat different to that of distributing recorded content on demand. A major difference is that delay is important to optimise (so that peers can view streams as "live" as possible) whereas throughput only needs to be large enough to view the stream (a peer cannot continue to download at faster than the rate the stream is broadcast).

A number of strategies might be considered for forming such topologies for live streaming. Minimising delay to the peercaster might be one strategy. Connecting to close (in terms of delay) nodes might be a related strategy. Another aspect to consider is whether

[*] richard@richardclegg.org
[†] dgriffin@ee.ucl.ac.uk
[‡] r.landa@ee.ucl.ac.uk
[§] e.mykoniate@ee.ucl.ac.uk
[¶] m.rio@ee.ucl.ac.uk
[||] Department of Electrical Engineering, University College London, London

it is important to aggressively minimise delay or closeness by making as many connections as possible to the lowest delay/closest node or whether it might be preferable to have a range of connections. The topologies formed are tested against several metrics which attempt to assess whether the topology is good at reducing delay, resilient in the case of peers dropping out and whether it ensures that the bandwidth is used fairly.

1.1 Background and related work

Distributing content over an overlay network has been the subject of numerous studies in recent years. Most of this research has been concentrated on non-live content where the emphasis is on increasing throughput rather than reducing delay. In early approaches like SpreadIt [4], a multicast tree is built by centralised logic running at the data source. Upon the arrival of a new node, the source is contacted to appoint an unsaturated node to be the parent of the new node. When the *smart-placement* policy is in effect, the parent node is also selected to be close to the new node, where proximity is inferred with traceroute messages. More recently, Bos [7] proposed a method which constructs a data distribution tree containing the *Euclidean Minimum Spanning Tree*, where the distance in the Euclidean space represents the network delay. A subset of stable and high capacity nodes are elected to become *super peers*. Super peers are interconnected to form a *Yao graph*, a structure which contains the Euclidean Minimum Spanning Tree. Normal peers attach directly to the closest super peer. The source routed multicast tree is built over the super peers topology based on the compass routing protocol.
The departure of a node in single distribution tree topologies results in complete loss of connectivity for all the nodes in the underlying subtree. To overcome this problem, several studies investigate streaming the data over a forest of multicast trees, each of which carries only part of the stream. Coopnet [8] is a forest-based streaming approach, where the authors identify a tradeoff between efficiency in terms of locality and path diversity required for resilience to node departures. Upon addition of a new node, the source returns a significantly large set of candidate parent nodes to ensure diversity. As an optimisation the candidate parent nodes are selected, similarly to SpreadIt, so that are they are *nearby* the newly added node.
Techniques for constructing trees typically assume global knowledge and at least one interaction with the source. Alternatively, overlay topologies can be constructed with local knowledge, where the connections are determined by each node and the data flow may take many alternative and potentially overlapping paths. In [9] a technique for clustering nodes to bins based on their locality is proposed. As a case study of this technique, the *BinShort-Long* overlay construction method is presented, where each node connects to $k/2$ randomly selected nodes from within its cluster (bin) and $k/2$ random nodes from anywhere in the system. A similar technique is proposed in [1] as an improvement for the BitTorrent protocol. The clustering here is done primarily to distinguish between nodes located in the same ISP, and nodes in different ISPs. Out of the total BitTorrent peers discovered by a new peer through the local tracker, all but k are selected to be local peers, with typical values 35 for total peers and 1 for the k external peers. This is done to reduce the traffic over the inter-domain links while still maintaining enough connections with external peers to receive the data. Finally, in [10] the authors formulate the *Minimum Delay Mesh problem* and prove that it is NP-hard. They propose a heuristic for constructing a shallow (low number of hops) and locality-aware (low delay at each hop) overlay topology. In order to minimise the number of hops, nodes with higher capacity need to be connected closer to the source.

The selection of the nodes to establish connections with, is done after calculating the *power* of each node, as a function of the node's locality and bandwidth availability.

2 Simulation methodology

In order to make the simulation of the overlay tractable it is necessary to abstract away the network itself and simulate only the overlay. The simulation described here makes as few assumptions as possible. It is assumed that each node has a fixed delay to every other node in the overlay (as described in the next section). It is also assumed that each node has a sufficient download bandwidth to obtain the entire stream and upload bandwidth to deliver a fixed proportion (which may be more than unity) of the stream.

2.1 Node distributions

Synthetic coordinate systems associate a coordinate with each peer in an overlay network, in such a way that the distance between the coordinates is a good estimate of some network property measured between the peers, predominantly *round trip time* (RTT). This can be achieved efficiently by using a limited set of end-to-end measurements to extrapolate those distances between nodes that were not explicitly measured. Thus, synthetic coordinate systems use a limited set of measurements to model the structural properties of the Internet, and then use this model to predict end-to-end properties (such as RTT) between arbitrary peers.
The first step in the operation of a network coordinates system is generating a *distance graph*, where links between peers represent distance measurements. This distance graph is then *embedded* onto a space that integrates some of the structural properties of the Internet. Examples of these include a standard Euclidean space [2], a Euclidean space augmented with a purely additive coordinate [3] or a hyperbolic space [11]. The embedding process can be viewed as an error minimization procedure where nodes are positioned in the space in such a way that the cumulative difference between the measurements and the embedded distances is minimized. Once this embedding has been done, and to the extent that the embedding space faithfully recovers the structure of the Internet for the measure in question, geodesic distances over this space are good predictors of the actual distances over the Internet [6]. This space will be referred to as *delay space*.
In the case of the simple simulation used in this paper, a standard two-dimension Euclidean delay space is used. Let N be the number of nodes in the system excluding the peercaster. The $N + 1$ nodes, numbered from 0 (the peercaster) to N are distributed over the two-dimensional Euclidean space. Each node has a co-ordinate (x_i, y_i) and the delay from node i to node j is obtained using the standard Euclidean distance from (x_i, y_i) to (x_j, y_j).
The question then becomes how to distribute the nodes on the delay space. For the purposes of this paper we use three generation methods to create random node distributions. In reality, nodes in an overlay network will cluster to some degree, for example, nodes in the real Internet are more prevalent in some areas of the world than others (clusters in large cities, particularly large cities with high levels of Internet usage). In the case of an overlay network based upon nodes wishing to download particular streaming content, the distribution will be further complicated by whether the content is of regional, national or global interest as well as what language the broadcast is in.

For this reason the simulation here is tested against different assumptions about how nodes might be randomly situated in delay space.
Flat node distribution $\mathcal{N}_F$: In this distribution the nodes are flatly distributed in a square delay space. For each node i, x_i and y_i are chosen randomly from a flat distribution in the interval $(-D, D)$. In the simulations given here $D = 0.25$ seconds (so the maximum delay between any two nodes is $\sqrt{2}/2$ secs).
Tightly clustered node distribution $\mathcal{N}_T$: This distribution simulates a situation where nodes are grouped into tight clusters. The following procedure is followed until sufficient nodes have been generated.

1. Coordinates position X, Y is chosen with a flat distribution where X and Y are chosen from the interval $(-D, D)$.
2. The position (X, Y) is modified by a small random perturbation (d_X, d_Y) where d_X and d_Y are chosen with a flat distribution in the interval $(-d, d)$.
3. Coordinate (X, Y) is recorded.
4. With probability p go to step 1, otherwise go to step 2.

In this distribution $D = 0.25$, $d = 0.005$ and $p = 0.01$.
Loosely clustered node distribution $\mathcal{N}_L$: This distribution is identical to the previous one but the clusters are more diffuse and contain fewer nodes $D = 0.25$, $d = 0.05$ and $p = 0.01$.
In each of the last two cases, after the distribution is created, the node order is randomised. Node order is important for local topology schemes (see section 3.2). This reordering prevents nodes being created in a convenient "by cluster" order with nodes locally close being created together.

2.2 Modelling assumptions

For simplicity it is assumed that each node attempts to download a stream as M separate and equally sized *substreams* – note, however, that this could also be thought of as simply an abstraction of, say, a chunk-based swarming system with M partners from whom equal amounts are downloaded. Assume that each node has capacity to download all M substreams and that nodes have upload capacities to upload only a limited number of substreams.
Each node has associated with it an upload capacity u_i which is the number of substreams it can support (for the purposes of bandwidth calculation each substream is considered to have a bandwidth of 1Mb/s – although the precise unit is unimportant and of the metrics described, only the bandwidth variance is affected by this). Note that it must be assumed that $u_0 \geq M$ (in order that all M substreams can be uploaded from the peercaster itself) and also for the system to scale it is important that $\overline{u_i} \geq M$ (the average peer has sufficient capacity to upload all M substreams). An implicit assumption is that system bottlenecks are only at the peers in the network. This may not always be the case in reality (for example several peers who belong to the same ISP may share access network capacity in the underlying network).
Nodes will then attempt to connect to at most M other peers in order to download the complete stream (nodes can download all M streams from a single partner node). A node i will accept at most u_i connections and request up to M connections. The complete set of connections will be referred to as a *topology* on the overlay network. This will be described in the next section.

For this paper u_i will be chosen from a random distribution. In addition u_0 will be fixed since it has such an important role in the network. The values used are $M = 4$ and u_i is chosen with equal probability from the set $\{1, 5, 10, 16\}$ – in this simulation no nodes are complete free-riders although some nodes can only produce 1/4 of a complete stream. The mean value of u_i is 8 so the system easily has capacity for every node to download the stream. As previously stated u_0 is a critical parameter in the system so $u_0 = 16$ for all simulations – the peercaster is always assumed to have a reasonable amount of bandwidth. This is to prevent the simulation results being greatly dependent on this single random selection (a simulation where $u_0 = 1$ might get very different results from one with $u_0 = 16$ even if all else was the same).

3 Topology generation and assessment

A topology on the network is a directed graph which may have more than one edge from node i to node j (an edge represents node i sending a single substream to node j). Let I_i be the number of incoming connections to node i and O_i be the number of outgoing connections to node i. A *valid topology* on the network is one where $I_i = M$ for $i \in 1, \ldots, N$, $I_0 = 0$ (every node apart from the peercaster is downloading a complete stream) and $O_i \leq u_i$ for all i (every node is within its upload bandwidth limit). In addition there must exist a walk from node 0 to every node i in the network (every node has a path to download from the peercaster).

3.1 Metrics

Because each node has M independent connections, variants on more usual network metrics are used here. For example, it is not simply the shortest path from a node to the peercaster to the node which is of interest but the path length along all paths.
The metrics listed in this session have been created with several considerations in mind. A “good” topology should have all or most of the following properties.

- Low delay to end nodes – this translates to nodes being able to view streams with good “liveness”.
- High resilience to churn – a peer-to-peer network is, by its nature, highly dynamic. The loss of any single node should not greatly affect the network.
- Diversity of paths – related to the above, an individual peer would want a diverse set of connections so that the loss of a single intermediate node will not affect every stream it is downloading.
- Equitable spread of bandwidth – the upload bandwidth requirements in carrying the stream is spread across all nodes rather than falling unduly on a small number of nodes.

Let $D_i(j)$ be the shortest path delay from node i to the peercaster assuming that the first “hop” is node i’s jth connection. Let $P_i(j)$ be the set of nodes which connect node i to the peercaster for its jth connection (not including i and the peercaster itself). If i connects directly to the peercaster for its jth connection then $P_i(j) = \emptyset$. Note that in pathological cases this shortest path may go back through the node i itself.
Let V_i be the vulnerability of node i to the removal of a single node. This is the number of streams connecting node i to the peercaster which could potentially be disrupted by

the removal of a single node (not including the peercaster). It is zero if and only if every node is directly connected to the peercaster. It is M if the shortest paths $D_i(k)$ go through one node (not including the peercaster) for all k.
Let S_i be the vulnerability of the system to the removal of node i. It is, in a sense, the dual of V_i. It is the total number of streams $D_j(k)$ (where $j \neq i$) which could be broken if node i were removed from the system.
Single figure metrics (the following metrics produce a single number).

1. Mean minimum delay $\mathbf{D}_{\min} = \sum_{i=1}^{N} \min_j D_i(j)/N$ – this is the mean of the minimum delay from a given node to the peercaster.
2. Mean maximum delay $\mathbf{D}_{\max} = \sum_{i=1}^{N} \max_j D_i(j)/N$ – this is the mean of the maximum shortest path from a given node to the peercaster.
3. Maximum system vulnerability $\mathbf{S} = \max_i S_i/MN$ – this is the proportion of routes which could potentially be damaged by the removal of a single node. It will be one if there is a single node (apart from the peercaster) which can disrupt every transmission path and zero if there are no nodes which can damage paths (only possible if every node connects directly to the peercaster). This measure is similar to finding the node with maximum Betweenness-Centrality [5]. It is a measure of a worst case vulnerability to churn in the network.
4. Mean node vulnerability $\mathbf{V} = \sum_{i=1}^{N}(V_i)/NM$ – this is mean of V_i over all nodes. It is a measure of how vulnerable the average node is to churn. The value is one if every node i can have all M streams broken along the shortest path by the removal of a single node not including i or the peercaster. The value is zero if and only if every node connects directly to the peercaster.
5. Bandwidth variance $\mathcal{B}_v = \text{var}\,(O_i)$ (for $i > 0$) – this attempts to assess whether the system load is split evenly between all nodes. If this is low then all nodes are using similar quantities of bandwidth but if it is high then some nodes are using large amounts of bandwidth and some little. Nodes with zero upload capacity are ignored by this metric.

3.2 Topology generation methods

The topology generation methods are split into two types: global and local fixed.
Global methods begin with all peer nodes present in the system. Connections can be chosen (at least potentially) with regard to every other node present in the network. Such methods are unrealistic in a large-scale real peer-to-peer system but may provide insight into the performance levels that can be expected from a topology. This can be thought of as a complete information godlike view of the system and its connections.
Local fixed methods have peers appearing on the network in sequence, peercaster first. Each peer chooses all M connections before the next peer joins the network. (The first peer present must connect M times with the peercaster). Such methods are practical for real peer-to-peer networks (although they may involve more information than could be easily obtained from a real peer-to-peer network).
An obvious extension to this would be local reevaluating methods in which nodes appear in a fixed order and initially get their connections using local methods but may subsequently change connections as new, better, nodes appear. These topologies will be studied in further work.

3.3 Global topology generation methods

Random global topology $\mathcal{TG}_R$ – this can be thought of as the base "worst" case for global topology generation. This first algorithm is described in detail despite its simplicity so that the reader can fully understand the implementation. Subsequent topology algorithms will be described fully but in a less verbose manner. Let S be the set of nodes which do not yet have M substreams $S := \{i \in 1, \ldots, N\}$. Let C be the set containing only the peercaster (node 0) $C := \{0\}$.

1. Randomly pick nodes i from C and j from S.
2. Make an outgoing connection from i to j. Set $I_j := I_j + 1$ and $O_i := O_i + 1$.
3. If $j \notin C$ then $C := C \cup \{j\}$.
4. If $O_i = u_i$ then $C := C \setminus \{i\}$.
5. If $I_j = M$ then $S := S \setminus \{j\}$.
6. If $C = \emptyset$ then an invalid topology has been reached. Begin the algorithm afresh.
7. If $S = \emptyset$ then all nodes have been connected. Otherwise go to step 1.

Closest first global topology $\mathcal{TG}_C$ – This algorithm attempts to minimise delays by connecting "close" nodes first. Let C be the set of nodes which are "connected" (can trace a route from the peercaster) and have capacity to spare (initially $C = \{0\}$). Each node i which does not yet have $I_i = M$ maintains a record of the node in the set $C \setminus \{i\}$ to which it has the smallest delay. The node n makes a connection to its closest node j. The node n is added to C if it is not already there. The node j is removed from C if its capacity is filled. The process continues until all nodes are connected with $I_i = M$
Small world $\mathcal{TG}_S$ – This topology has $M - 1$ connections picked according to the rules in $\mathcal{TG}_C$ and one node picked randomly according to the rules in $\mathcal{TG}_R$. The idea being to form a small world type network with mainly local connections but one global connection.

3.4 Local fixed topology generation methods

Local random $\mathcal{TLF}_R$ – In this algorithm, the nodes appear in order and connect to random nodes which have spare capacity. As it appears a node picks one node from the set of nodes with spare capacity at random and connect to it. This process is repeated until the node has M connections.
Local closest first $\mathcal{TLF}_{C1}$ – In this algorithm, the nodes appear in order and each node, as it appears, makes connections to the closest node to it (in terms of delay) until either all M connections are made or that node runs out of upload capacity (in which case the next closest node is used and so on).
Local closest with diversity $\mathcal{TLF}_{C2}$ – This topology is the same as $\mathcal{TLF}_{C1}$ but the nodes attempt to connect to M different other nodes if possible, if no such nodes are available then the nodes connect to the closest node with which they only have one connection and so on. (For example the first node to appear must by necessity make all M connections to the peercaster itself).
Local minimum delay first $\mathcal{TLF}_{D1}$ – This topology is formed by each node connecting to that node with capacity to spare from which it can get the lowest shortest path delay (in the same way as $\mathcal{TLF}_{C1}$ did).

Local minimum delay with diversity $\mathcal{TLF}_{D2}$ – This topology is the same as $\mathcal{TLF}_{D1}$ but with the modification that the node attempts to connect to M different nodes if possible.
Local small world $\mathcal{TLF}_S$ – This topology has $M-1$ nodes picked according to $\mathcal{TLF}_{C2}$ and one node picked randomly according to $\mathcal{TLF}_R$.

3.5 Criticisms of the metrics and topologies

While the selected metrics are intended to be proxies for the properties declared at the start of this section, it is important to remember that they do not have a simple and direct relationship with those properties. For example, the delay based metrics are intended as a proxy for the the delay from the peercaster to the end user. However, the actual delay experienced by the user will not directly relate to the metrics specified here. In a substreaming system for example, this would depend upon exactly which nodes that particular substream travelled through. This itself might be subject to optimisation separately from topology. It would also depend upon exactly where in the stream the particular peer chose to upload from. The delay would also depend on choices made by the peer since there would be a trade off between delay and reliability in such a system (a peer may decide to allow a time buffer to allow for future jitter in downloading a stream).
Because the focus of this paper is on the effects of topology rather than on the effects of packet scheduling and buffering strategies it was decided to use simpler simulations and metrics which are only proxies for the ideal measurement.
The topologies considered are only a small subset of the possible topology construction methods. A moment's consideration will come up with many more possibilities, however, the number of results presented in this paper is already large. Future work will investigate different topologies.
Of the topologies considered, an obvious criticism is that they all have complete and accurate knowledge of the system. This approach is taken because the aim of this work is to develop an optimal strategy for real peers to implement. In the case of the local topologies this is the system "to date" and in the case of the global topologies this is the entire universe of peers. No "churn" is accounted for, nodes only enter. Both topologies have potential problems with the possibility of connecting up unrealistic or impossible networks. In the case of the global topologies, topologies can be generated with a minimum cut less than the stream bandwidth. This problem will be addressed in subsequent work. In the case of local topologies it is possible that the nodes could appear in an order which made connecting the network impossible. For example, with $M = 4$, if nodes with $u_0 = 4$, $u_1 = 1$ appeared then node 2 could not find four upload connections. This problem is addressed by ensuring that the node order is such that valid local topologies are always possible.

4 Results

Figure 1 shows the effects of the node distribution algorithms. The scale is delay in milliseconds.
It is useful to show an example of topology creation. A simple situation with ten nodes is shown with the following upload capacities (beginning with the peercaster, node 0) $(16, 1, 5, 5, 10, 10, 10, 16, 1, 5)$.

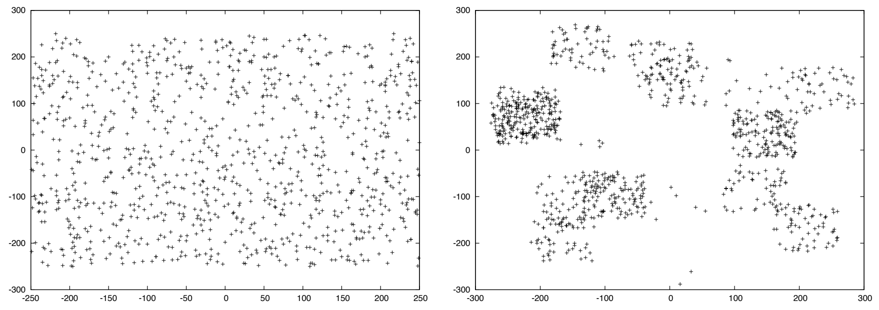

Figure 1: Flat $\mathcal{N}_F$ (left) and loosely clustered $\mathcal{N}_L$ (right) node distributions of 1000 nodes.

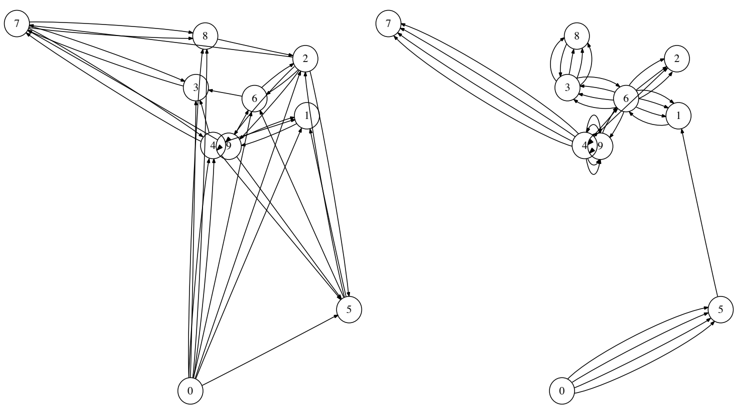

Figure 2: Ten nodes connected using the topologies global random $\mathcal{TG}_R$ (left) and global closest $\mathcal{TG}_C$ (right).

Figure 2 shows two of the global topologies, node 0 is the peercaster. The arrows indicate the direction in which data is transmitted. As might be expected the global closest topology $\mathcal{TG}_C$ has many node pairs with a number of links in common. For example, all the upload links to node 5 come from the same node (the peercaster). This is because the first connection made is from node 0 to node 5 and after this connection node 5 is still the closest node to the set of connected nodes. Node 1 can only supply one upload stream because of its bandwidth constraints. In this topology nodes 1 and 6 mutually upload (this is entirely possible in a live streaming system). It should also be noted that the $\mathcal{TG}_C$ topology is unrealistic in this case. The single link from node 5 to node 1 is a bottleneck in the network. Topology $\mathcal{TG}_R$ as might be expected has much more diversity in connections.

Figure 3 shows the two local topologies $\mathcal{TLF}_{C1}$ and $\mathcal{TLF}_{C2}$. As can be seen, in $\mathcal{TLF}_{C1}$ it is common for a pair of nodes to have several connections. Indeed this pattern is only broken because of nodes running out of upload capacity (for example node 1). In topology $\mathcal{TLF}_{C2}$ close connections are still made but nodes choose diverse connections. Because of the nature of the local topologies, node 1 has to connect to

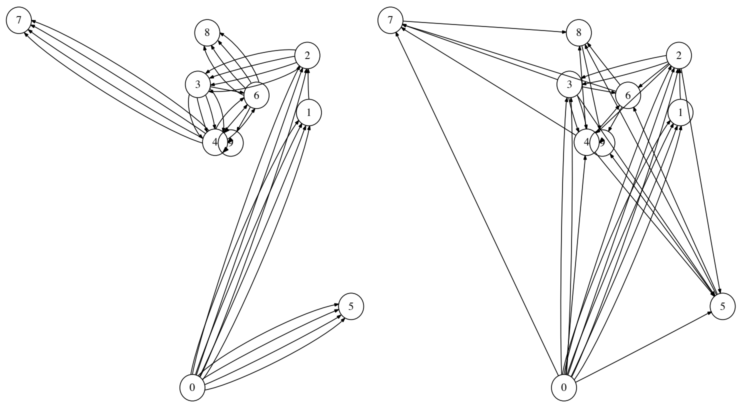

Figure 3: Ten nodes connected using the local closest topologies $\mathcal{TLF}_{C1}$ (left) and $\mathcal{TLF}_{C2}$ (right).

node 0 only (because at this point only node 1 and node 0 are in the system). Similarly node 2 can only connect to node 1 or node 0. The fact that each node has full upload bandwidth before other nodes can connect to it ensures that no unrealistic topologies (in the sense of the previous section) can be constructed using the local topology methods.

4.1 Experiments performed

For one of the node distributions $\mathcal{N}_F$, $\mathcal{N}_L$ and $\mathcal{N}_T$ 10,000 nodes are generated. Topologies are constructed from 100, 200, 500, 1,000, 2,000, 5,000 and 10,000 of these nodes (the nodes randomly chosen from the 10,000). To check the stability of each metric, each set of topology, node distribution and number of nodes is repeated ten times. The mean value of each of the metrics is calculated for the set of ten experiments and a 95% confidence interval is also obtained.
The majority of the results are presented with error bars representing the 95% confidence intervals. Note also that small offsets are introduced in the x-axis in order that the error bars do not overlap.

4.2 Delay metrics

Figure 4 shows $\mathbf{D}_{\max}$ for the topologies on the flat node distribution $\mathcal{N}_F$. Of the global topologies, $\mathcal{TG}_R$ is, as might be expected, the worst performing and scales badly with max delay increasing rapidly with the number of nodes in the system. $\mathcal{TG}_C$ performs better but still does not scale well. $\mathcal{TG}_S$ is the best performing and its scaling seems extremely good, indeed surprisingly so.
In the case of the local topologies, figure 4 shows that $\mathcal{TLF}_{C2}$ has the lowest delay, with $\mathcal{TLF}_S$, $\mathcal{TLF}_{C1}$ and $\mathcal{TLF}_{D2}$ also performing well. Bad performers are $\mathcal{TLF}_R$ (the random local topology) and, perhaps more surprisingly $\mathcal{TLF}_{D1}$. A possible explanation for the latter is that each node as it arrives attempts greedily to get its lowest delay to the peercaster. The earliest nodes to arrive connect directly to the peercaster. Once the peercaster bandwidth is exhausted a second wave of nodes has a high likelihood of connecting to any of those nodes close to the peercaster. Low delay nodes

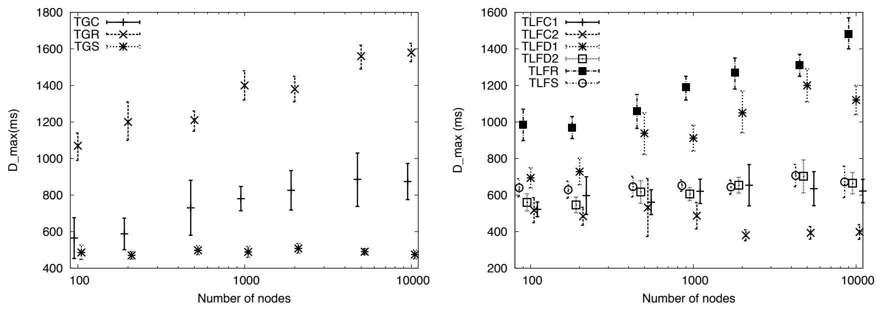

Figure 4: $\mathbf{D}_{\max}$ for global (left) and local topologies (right) on a flat node distribution $\mathcal{N}_F$.

would quickly find their upload bandwidth used up. The $\mathcal{TLF}_{D1}$ topology reduces this effect somewhat by insisting that nodes upload from a diverse selection of other nodes.

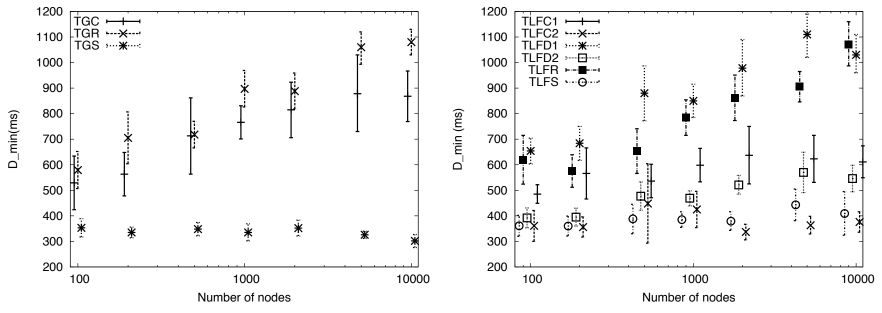

Figure 5: $\mathbf{D}_{\min}$ for global (left) and local topologies (right) on a flat node distribution $\mathcal{N}_F$.

Figure 5 shows the same experiment as figure 4 but uses the minimum delay metric $\mathbf{D}_{\min}$. The graphs are essentially similar but $\mathcal{TG}_R$ and $\mathcal{TLF}_R$ perform better than previously. This is almost certainly a result of the fact that their random nature is likely to make the worst connection much worse than the best connection they experience. For the local topologies, $\mathcal{TLF}_S$, $\mathcal{TLF}_{D2}$ and $\mathcal{TLF}_{C2}$ are the best performing.
Figure 6 shows the same results as figure 5 but with the tightly clustered node distribution $\mathcal{N}_T$. The differences between the graphs are not too great. The clustering seems to lower the minimum delay for most measures, perhaps because there are always some nodes extremely close to the peercaster. The only policy which appears significantly affected is $\mathcal{TG}_C$ which performs consistently much better in the presence of clustering. Note especially that the topology has changed character from dramatically increased delay as the number of nodes increases to almost no increase in delay as the number of nodes increased. This is explored further in section 4.6.

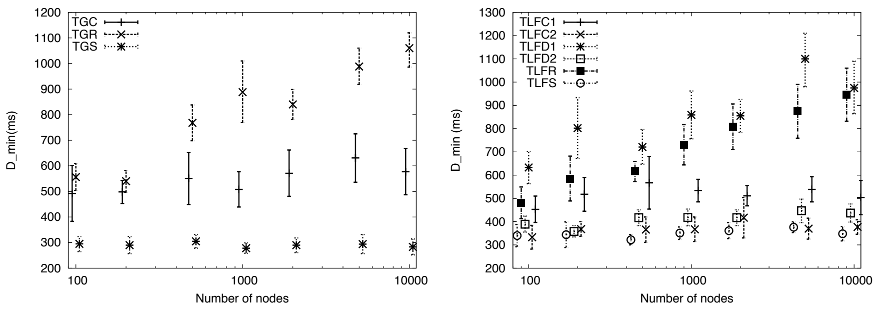

Figure 6: $\mathbf{D}_{\min}$ for global (left) and local topologies (right) on a tightly clustered node distribution $\mathcal{N}_T$.

4.3 Vulnerability metrics

Two vulnerability metrics were defined in section 3.1. The first **S** attempts to assess the vulnerability of the system to the removal of a single node. The second **V** attempts to measure the vulnerability of the average node to the removal of some other node from the system.

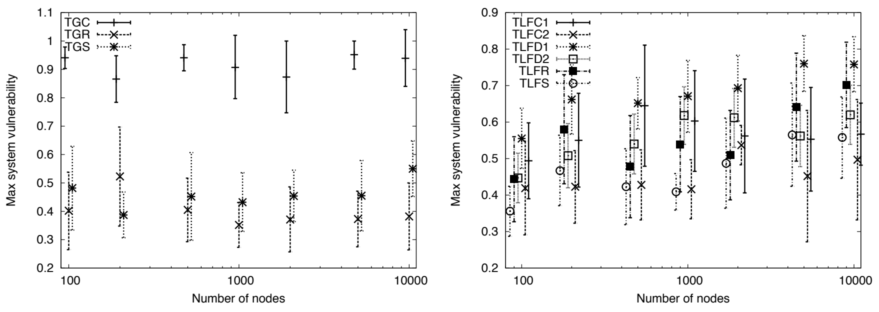

Figure 7: Maximum system vulnerability **S** for global (left) and local topologies (right) on a tightly clustered node distribution $\mathcal{N}_T$.

Figure 7 shows the system vulnerability to node removal for various topologies. Of the global topologies the $\mathcal{TG}_C$ is by far the most vulnerable. The random topology $\mathcal{TG}_R$ is by comparison very resilient although the small world topology $\mathcal{TG}_S$ is almost as resilient. For the local topologies the picture is less clear. The delay based topology $\mathcal{TLF}_{D1}$ is the least resilient. This is because this topology will tend to arrange itself along low delay trees. The introduction of a requirement for diversity in connections in $\mathcal{TLF}_{D2}$ reduces this vulnerability somewhat. It should be noted that most of the confidence intervals in this graph are very large. The system vulnerability can change greatly with each run using the same parameters.
Figure 8 shows similar but not identical results using the metric $\mathbf{V}$ which attempts to measure how vulnerable each node is to the removal of single nodes from the network. In this case, $\mathcal{TG}_C$ performs terribly as does $\mathcal{TLF}_{C1}$, $\mathcal{TLF}_{D1}$ and to a lesser extent $\mathcal{TLF}_{C2}$. The random topologies ($\mathcal{TG}_R$ and $\mathcal{TLF}_R$) obviously do well as these are

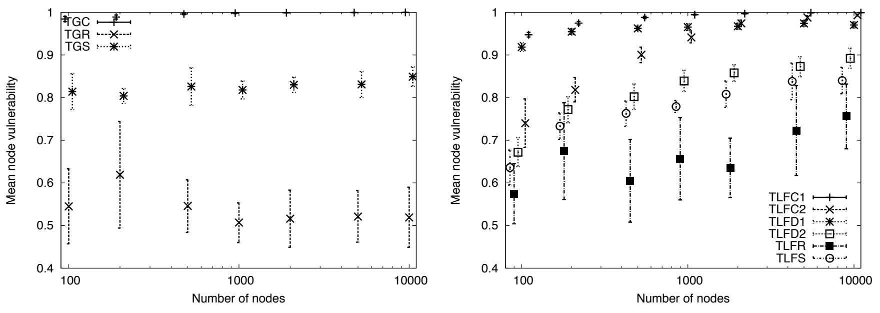

Figure 8: Mean node vulnerability **V** for global (left) and local topologies (right) on a tightly clustered node distribution $\mathcal{N}_T$.

unlikely to have single points of failure. The small world topologies $\mathcal{TLF}_S$ and $\mathcal{TG}_S$ also perform relatively well, probably due to their random elements.

4.4 Fairness metric

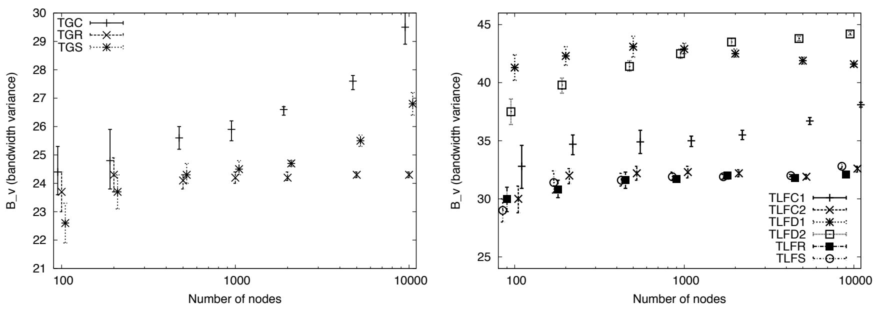

Figure 9: Bandwidth variance $\mathcal{B}_v$ for global (left) and local topologies (right) on a tightly clustered node distribution $\mathcal{N}_T$.

The results on bandwidth variance are shown in figure 9. A low variance would indicate nodes sharing upload capacities “fairly” (although this does not account for node capacities – no clear picture emerged when metric involving proportional bandwidth was used). The topology $\mathcal{TG}_C$ has a higher bandwidth variance than the other global topologies. However, the worst performing topologies are clearly $\mathcal{TLF}_{D1}$ and $\mathcal{TLF}_{D2}$. This is because those topologies are very likely to efficiently exploit peers with low delay connections to the peercaster and such peers are likely to have their entire bandwidth exhausted.

4.5 Tradeoffs in metrics

Figure 10 shows how vulnerability interacts with delay. The plot on the left is of **V** versus $\mathbf{D}_{\max}$ and on the right is **S** versus $\mathbf{D}_{\max}$. Each point on the plot is for the mean of ten runs for a given topology and a given node distribution. As can be seen, the

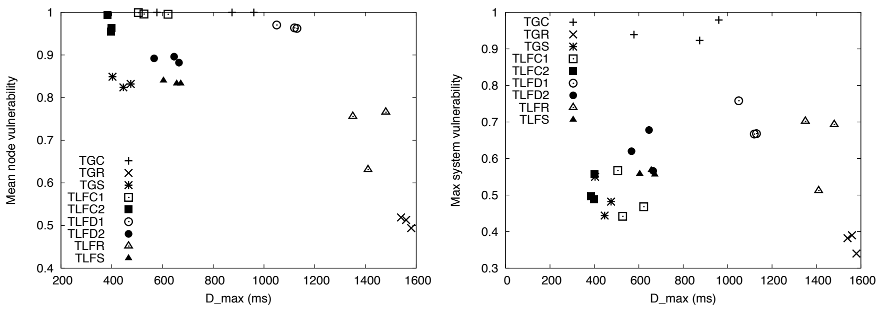

Figure 10: Mean node vulnerability versus **V** versus $\mathbf{D}_{\max}$ (left) and max system vulnerability **S** versus $\mathbf{D}_{\max}$ (right) for all topologies (using all three node distributions).

results are not particularly sensitive to the node distribution used apart from for the topology $\mathcal{TG}_C$.

The best performing topologies are those to the bottom left of the graphs. For the system vulnerability measure **S** then the best topologies seem to be $\mathcal{TLF}_{C2}$ and $\mathcal{TG}_S$ and to a lesser extent $\mathcal{TLF}_S$ and TLF_{C1}. For the node vulnerability measure **V** then the best topologies seem to be $\mathcal{TG}_S$ or, if resilience is very much more important than delay, $\mathcal{TG}_R$. Of the local topologies, $\mathcal{TLF}_S$, $\mathcal{TLF}_{D2}$ and $\mathcal{TLF}_{C2}$ perform best with $\mathcal{TLF}_R$ being preferred only if delay is much less important than network resilience.

4.6 The effects of the node distribution

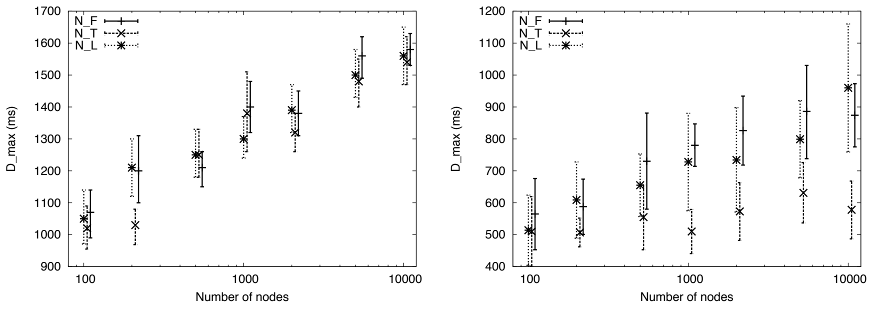

Figure 11: Maximum delay $\mathbf{D}_{\max}$ versus number of iterations for topology $\mathcal{TG}_R$ (left) and TG_C (right) showing the effects of node distributions $\mathcal{N}_F$, $\mathcal{N}_T$ and $\mathcal{N}_L$.

Figure 11 shows the effect of node distribution on delays. The plots are of the number of nodes versus the maximum delay $\mathbf{D}_{\max}$ for a given topology. Each line represents one of the three methods for creating a node distribution $\mathcal{N}_F$, $\mathcal{N}_T$ and $\mathcal{N}_L$ (a flat distribution, a tightly clustered distribution and a loosely clustered distribution). The differences between the distributions (as seen in figure 1) is quite marked. As can be seen, for the random distribution $\mathcal{TG}_R$ the impacts of the node distributions on the metrics chosen were minimal (in almost all cases the 95% confidence intervals overlap). This seems to be a fairly typical result. This can be seen by the fact that in figure 10 the three points for a given topology are almost always clustered together indicating that the topology is of much greater importance than the choice of node distribution used. The exception is for the $\mathcal{TG}_C$ topology. This topology is its closest connected neighbour from all the nodes which will ever enter the system. This seems to be the only case where node clustering makes a marked difference to the results used. However, it should be noticed that even in this case the 95% confidence intervals overlap for most experiments.

5 Conclusions and further work

It is clear that much work remains to be done on this topic, however, some clear conclusions can be drawn. Naive policies like "connect to my closest peer" are not as effective as might be thought in reducing delay to the peercaster. This can produce systems which have a high delay from the peercaster to the peer and also with a high vulnerability to churn. This problem can be mitigated by the so-called "small-world" topologies used here where most connections are local but some are distant.
For the delay measures investigated, some of the topology methods used seem to scale very well indeed with system size. The global small world topology seemed almost delay invariant as the number of nodes in the system increased. Of the local topologies, the topology which tried "hardest" to minimise delay (with nodes aggressively using all the bandwidth of nodes to which they had the lowest delay connection to the peercaster) fared surprisingly badly both in terms of delay and in terms of vulnerability.
With global system knowledge the small world topology (three close connections and one random) performed extremely well in terms of both delay and vulnerability. For the local topology policies the local small world policy or the local closest with diversity policy seemed to offer the best trade off.
An interesting outcome of this research is that, for the parameters used here, the system seemed extremely insensitive to the node distribution used. The node distribution policies were chosen so that the nodes were laid out in a delay space of approximately the same size. However, only for the global closest topology policy were significant differences found in metrics due to a change in the node distribution. This is important since, if this conclusion is more widely applicable, it could free modellers from the (possibly extremely time consuming) task of attempting to validate a peer-to-peer model against a realistic distribution of global delay.
Much remains to be done to complete this work and there are many further avenues which could be investigated. The metrics used could be improved (although at the expense of computational complexity). The global topology methods used could sometimes generate unrealistic topologies (in the sense that a max-flow/min-cut would find that the network had a max flow less than the stream total bandwidth). This could be improved by adding constraints on the global topology creation (a node could only have n uploads if it already had n downloads and had capacity for at least n uploads). There are many other simulation parameters which could be investigated. The choice of four streams here and the distribution of upload capacities was somewhat arbitrary. However, it is difficult to run simulations with too many "degrees of freedom". A repeated experiment with only one node distribution topology but differences in the distributions of upload bandwidths might generate some interesting results. Indeed a large problem with this research is that the state space to explore is extremely large even in this simple simulation.
An obvious next stage is to allow the local topologies a "reevaluation" stage. That is

to say, that after peers had chosen their upload connections, a second selection process would allow peers to swap to a different uploader. This adds complications since it might be desirable to allow a peer to “displace” another downloader, however, this might create problems if the displacement caused other problems with the topology (for example by cutting off sections of the network from the peercaster).

References

[1] R. Bindal, Pei Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 66–66, 2006.

[2] Russ Cox, Frank Dabek, Frans Kaashoek, Jinyang Li, and Robert Morris. Practical, distributed network coordinates. *SIGCOMM Comput. Commun. Rev.*, 34(1):113–118, 2004.

[3] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: a decentralized network coordinate system. *SIGCOMM Comput. Commun. Rev.*, 34(4):15–26, 2004.

[4] Hrishikesh Deshpande, Mayank Bawa, and Hector Garcia-Molina. Streaming live media over a peer-to-peer network. Technical Report Stanford Database Group 2001-20, CS Department, Stanford University, August 2001.

[5] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.

[6] Jonathan Ledlie, Paul Gardner, and Margo I. Seltzer. Network coordinates in the wild. In *NSDI*. USENIX, 2007.

[7] Eng Keong Lua and Xiaoming Zhou. Bos: Massive scale network-aware geometric overlay multicast streaming network. *Global Telecommunications Conference, 2007. GLOBECOM ’07. IEEE*, pages 253–258, Nov. 2007.

[8] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking, 2002.

[9] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection, 2002.

[10] D. Ren, Y.-T. Li, and S.-H. Chan. On reducing mesh delay for peer-to-peer live streaming. *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1058–1066, April 2008.

[11] Yuval Shavitt and Tomer Tankel. The curvature of the Internet and its usage for overlay construction and distance estimation. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages – 384. Proc. of IEEE Infocom, April 2004., 2004.