Cost-Efficient Peer-to-Peer Topologies

Benjamin Stewart Adrien Treuille

December 9, 2002

1 Introduction

Currently, peer-to-peer (P2P) overlay networks such as Gnutella are costly for Internet Service Providers (ISPs) to support. The problem is that *connections between hosts in the overlay are established without respect for the underlying Internet topology.* Hosts across the world are just as likely to be connected as hosts right next to each other (Figure 1).

While the user is insulated from the cost of such an overlay topology, the ISP must pay dearly for this traffic. The ISP faces a two-tiered pricing system:

1. Hosts on the same ISP network (or *peering group*) are called *local* to one another. Traffic between them is essentially costless.
2. Hosts on different ISP networks are called *global* to each other. Traffic between them costs the ISP money.

This state of affairs sets up a situation in which ISPs have an incentive to block P2P traffic to global hosts in order to reduce the considerable costs they incur. For an example of this, see [3].

Thus, it would be beneficial to have a P2P network topology that respects the Internet's cost structure. Specifically, we would like to see a P2P overlay network topology that has as few connections between global neighbors as possible, while maintaining many local connections (Figure 2). Of course, we should not get rid of all global connections in the overlay: this would partition the network. Instead, we would like to minimize the number of global connections needed to maintain a single connected network.

We show how to achieve this goal in the context of the Gnutella network. We designed and implemented a Gnutella client called UWtella that induces a topology with many less costly links compared to an ordinary Gnutella topology. Our solution has a number of benefits:

- Connectivity decisions need only be made at each Gnutella client. No overarching control structure is needed.

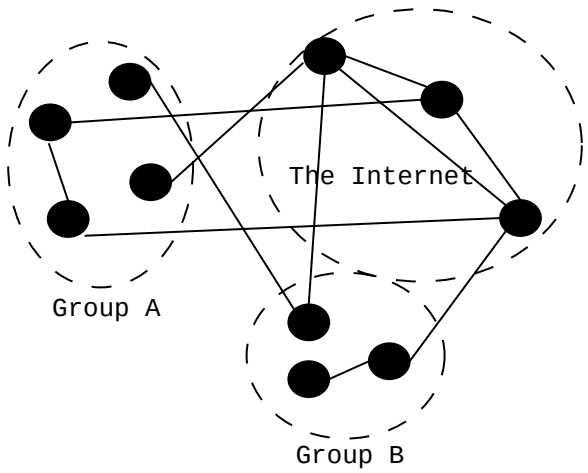

Figure 1: Depiction of an overlay network topology. A and B are ISP networks.

- We do not change or add to the Gnutella protocol.
- Our system requires minimal cooperation from the ISP. This is important because ISP may want to reduce costs without acknowledging possibly illegal activity on the P2P network.
- Our technique is not limited to Gnutella. It could be applicable to many self-organizing overlays.

Note that we do not address the problem of reducing *file transfer* traffic on the Gnutella network. File transfers are accomplished "out of band" by direct connection between Gnutella clients. Therefore, they are difficult to control. Instead we show how to induce a cost-effective topology on the Gnutella network itself. This means that "in band" Gnutella traffic, such as administrative messages and file queries travel mostly along costless links.

The rest of the paper is organized as follows: Section 2 discusses related work on peer-to-peer networks, Section 3 gives an overview of the Gnutella protocol, Section 4 explains our approach, Section 5 outlines the methodology of the experiments run in this study, Section 6 presents results from these experiments and Section 7 concludes with suggestions for future work.

2 Related Work

The corpus of research on peer-to-peer file sharing networks is limited because programs such as Gnutella are a fairly recent phenomenon. Much of the research focused on characterizing the traffic on Gnutella networks.

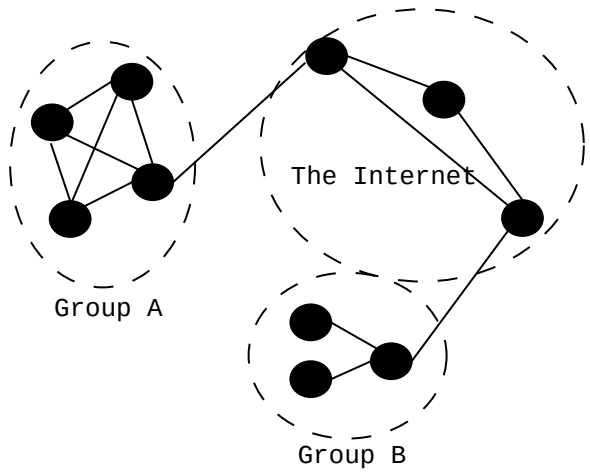

Figure 2: Depiction of the overlay topology we wish to induce.

In [7], Matei studied the Gnutella network's topology and patterns of traffic. Matei discovered that a certain type of Gnutella message, Pings, make up more than 50% of the traffic on the network. This is a result of the administrative aspects of the messages and flooding routing scheme used to propagate the messages. Various parties in academia and on the Internet propose improvements to Gnutella protocol to reduce the overabundant Gnutella messages. These solutions typically involve reasoning about when to send a messages, message caching, and message multiplexing. Examples of such revisions are [5] and [8].

To address overabundant messages, Matei suggests that a third-party agent monitor the topology and make requests for Gnutella clients to drop or add links as necessary to keep network topology optimal. If one considers minimal global neighbors as a criterion for optimality, then this approach could create cost-effective network topologies.

In [9] Saroiu et. al. outlined techniques they used for studying peer-to-peer file sharing networks. The authors devised tools to externally measure a collection of peer-to-peer file sharing network characteristics: network topology, network volume, bottlenecks and content distribution. Much of Sariou et. al.'s work in [9] provided inspiration for the work outlined in this paper. In particular, the diagrams of Gnutella topology made us realize that if clients discriminate in connecting with one another then the topology can be improved.

Several people have studied peer-to-peer content distribution. In a related study to [9], Saroiu et. al. analyzed Internet Content Delivery Systems in [10]. For peer-to-peer programs such as Gnutella and Kazaa, the study measured the amount and type of data exported and imported into the UW network. This

study concluded that the UW network exported much more data than it imported, acting like a data hub. Also, it was shown that a small number of servers took the majority of the burden of the workload in the peer-to-peer overlay network, which is against the ideology of peer-to-peer systems to spread the workload for scalability.

Adar and Huberman in [1] studied user traffic on Gnutella network and concluded that there is quite a bit of "free riding." After observing Gnutella traffic over a 24-hour. period, Adar and Huberman discovered that 70% of the Gnutella users monitored share no files and nearly 50% of all query responses were returned by the top 1% of sharing hosts.

Projects such as CHORD [11], FreeNet [2] and CAN [6] focus on building distributed systems based on peer-to-peer ideas. Most of theses systems look to place content at strategic locations in the network in order to preserve the capabilities of the system's indexing scheme.

The Gnutella spec [4] offered a detailed description of the protocol itself, which we used to develop our own client.

3 Overview of Gnutella

Before we delve into our technique, we offer a brief overview of Gnutella. Gnutella is a P2P file-sharing protocol. The protocol has been implemented by a number of different vendors and open-source projects.

When a Gnutella clients starts, it learns about the location of other clients through some means. Client discovery is not part of the protocol. Usually, a Gnutella implementation ships with a seed list of several thousand clients that have been observed in the past. This list initializes the client's cache which is continuously updated with new hosts once the client has connected to the network. In more recent implementations, clients will connect to public repositories of recently observed clients in order to jumpstart the connection process.

A connected Gnutella client has a number of *Gnutella connections* to other clients called its *neighbors*. Gnutella connections are TCP connections that remain open until one of the clients shuts down, or otherwise decides to close the link. Packets called *descriptors* pass over the Gnutella connections. Descriptors come in several varieties. The four most important are:

- *Pings* are sent out to verify connectivity and to discover new hosts. Clients either reply to incoming pings or pass them on to their neighbors.
- *Pongs* are sent in reply to pings. They indicate the presence of a host, and contain information about the connectivity, speed, and file database size of that host.

- *Queries* are sent to find files by keyword. Incoming queries are tested against the local file database, and then passed on to other neighbors. In this way, queries flood the network.
- *Query Hits* are routed back to the querying client. They indicate the presence of some file in the local database matches a keyword in the corresponding query.

Pings and queries flood the network, while their counterparts, pongs and query hits, are routed back to the originator of the corresponding descriptor. All Gnutella descriptors have a TTL field which controls the flooding, and assures that no descriptors travel forever.

When a user discovers a file she wishes to download, she may indicate so to her Gnutella implementation. At that point, a separate "out of band" HTTP connection is established between the downloader and the downloadee to transfer the file. For a detailed, technical description of the Gnutella protocol, see [4].

In general, Gnutella clients are "greedy" with respect to connecting to other clients. They immediately connect to available clients, and accept all incoming Gnutella connections until a certain threshold of connections is met. This paper shows that if clients exercise some discretion over with whom they connect, then far less costly Gnutella topologies emerge.

4 Solution

We have implemented a fully functional graphical Gnutella client called UWtella (Figure 3) that includes all the standard Gnutella features, including interoperability with third-party Gnutella clients, file uploads and downloads, and querying across the Gnutella network. Our main contribution is that, unlike other Gnutella implementations, UWtella has two special functions:

```
boolean isLocal(IpAddress x);
```

This function returns true if and only if x is on the zero-cost side of the network. Semantically, this function should partition the Internet into zero-cost ISP networks (called *peering groups*). In our experiments UWtella used a hard-coded list of local IP addresses to compute `isLocal()`. However, in the future, we envision that the ISP will provide this information to its customers, perhaps in the form of a web-based list.

Note that this information is of general interest. So in providing this information, the ISP is not specifically acknowledging any P2P applications. Also, it is in the interest of the ISP to provide this information as it can be used to reduce P2P traffic on its network.

UWtella's second special function is:

Figure 3: Screenshot of UWtella

```
boolean probablyHasGlobalConnectivity(Neighbor n);
```

This function has the following semantics: Let n be one of our neighbors, that is, a Gnutella client with which we are connected. If n recently relayed a descriptor bearing a non-local IP address (determined by `isLocal()`), then `probablyHasGlobalConnectivity(n)` is true, otherwise it is false.

Given these two functions, UWtella has a thread that tries to retain global connectivity, either by connecting to local neighbors who have global connectivity, or, failing that, by directly connecting to a non-local client. Pseudocode for this thread is given below:

```
periodically {
 if (none of my neigbors probably have global connectivity)
      try to connect to a global neighbor;
}
```

Each UWtella client randomizes the interval of time before checking for global connectivity. This is to decrease the probability that clients in the same ISP perform connectivity checks at the same time and try to connect to global neighbors simultaneously.

These two functions in conjunction with the thread that tries to maintain global connectivity are referred to as UWtella's *connectivity logic*. Our results show that a network of UWtella clients implementing this logic typically exhibits

topologies in which there are very few, but at least one, link between peering groups. The resulting topology is both qualitatively and quantitatively superior to the topologies that emerge without this connectivity logic enabled.

5 Experimental Methodology

We performed eight experiments, each ten times. In each experiment we do the following:

1. Choose the *number of peering groups* to simulate, the *number of clients* in each peering group, and the *maximum number of Gnutella connections* each client is allowed to have. We also choose *whether or not to activate our connectivity logic*. Experiments performed without connectivity logic are used as control.

2. We run a script that launches a number of UWtella clients across the UW computer science network. All clients in the same peering group are launched on the same computer, whereas different peering groups run on different computers. All clients are initialized with a Gnutella address cache consisting of all other clients in the experiment, so they know to connect to one another. To avoid degenerate topologies, we randomize each client's cache before deploying the UWtella clients.

3. The clients automatically attempt to connect to one another, either using our connectivity logic or not, depending on the experiment. As part of the connection process, and to establish global connectivity, all clients periodically flood the network with Ping descriptors (and reply with Pongs).

4. As the clients send and receive descriptors, they log the number of bytes sent across each link into trace files.

5. Simultaneously, we run another script that takes snapshots of the Gnutella topology at various points in time.

6. We then close down the network, and run postprocessing tools on the trace and topology data.

We ran our experiments in a number of different configurations to probe different aspects of our system. For each configuration, we experimented both with and without our connectivity logic. The configurations are:

1. *A network consisting of 4 peering groups, each peering group having 3 clients, and each client being allowed 4 Gnutella connections.* This was the smallest experiment, consisting of just twelve nodes. We used this configuration as a debugging tool. Also, we found that our system was more likely to create degenerate topologies (such as partitions) when there were few nodes. So these experiments allowed us to probe such degeneracies.

2. *A network consisting of 9 peering groups, each peering group having 9 clients, and each client being allowed 11 Gnutella connections.* This was a larger experiment. Note that the minimum number of Gnutella connections needed for each client to form a clique within its peering group is 8. Giving each client 9 connections would have allowed them additionally to create one global connection. By giving each client 11 connections, we were, in effect giving the clients "too many" connections. This allowed us to demonstrate that our system (correctly) does not take advantage of all these connections.

3. *A network consisting of 16 peering groups, each peering group having 5 clients, and each client being allowed 5 Gnutella connections.* Note that this experiment has 80 nodes, while the last has 81. This experiment allowed us to see what happens when we increase the number of groups, while holding the number of clients roughly constant. Note also that every client has exactly enough connections to form a clique within the peering group while leaving one connection open for global connectivity.

4. *A network consisting of 16 peering groups, each peering group having 5 clients, and each client being allowed 7 Gnutella connections.* This experiment was the same as that above except that we allowed each client more connections. Once again, we were trying to show that our system will not use these connections once global connectivity has been established.

In our results section, we show pictures of the resulting topologies as well as quantitative data from each of these experiments. In our future works section we discuss some limits of the experimental methodology, and describe how we could make these experiments more robust in the future.

6 Results

Our results show that Gnutella networks employing our connectivity logic are both qualitatively and quantitatively better than ordinary Gnutella networks.

On a qualitative level, pictures of the topologies resulting from experiments without our connectivity logic exhibit a plethora of red lines, indicating costly global connections (Figures 4, 6, 8, and 10). Compare this with the experiments in which we enabled our connectivity logic. (Figures 5, 7, 9, and 11). In these experiments there are mostly costless green lines, and only few red lines are maintained to keep the network from partitioning.

A minimally connected network topology would include a connected graph of green lines within each of the N groups and $N-1$ red lines spanning the N groups in order to maintain connectivity amongst all clients in the network. Though the experiments which utilized our connectivity logic are not minimal, the topologies are an improvement over the topologies created without using

our connectivity logic. Moreover, it is not clear if "minimality," in this sense, is better: a Gnutella overlay that merely spans its nodes runs the risk of being partitioned if a linchpin node goes down.

In the experiment involving 4 groups, 3 clients per group and each client was allowed to 4 Gnutella connections the connectivity logic was able to reduce the number of red lines but partitioned the network (Figure 5). This is in part due to the existence of double connections, i.e. two neighbors possessing incoming and outgoing connections with each other. In group C, each client has a double connection with its neighbors. This prohibits each client from establishing a connection with a global neighbor. We discuss this UWtella bug in the Future Work section.

A positive aspect of our technique can be seen by comparing the results of the final two experiments which employed our connectivity logic. These experiments featured configurations with 16 groups and 5 clients per group. Thus, in order for a client to connect to each of its local neighbors, 4 Gnutella connections must be used. Figures 11 and 9 show the topologies generated in these experiments. In the later figure, each client was allowed to 7 Gnutella connections. In the former, that number was 5. In both figures there are relatively the same number of red lines. This demonstrates that clients did not use all of the possible Gnutella connections to connect to global neighbors in the second experiment. Thus, our connectivity logic enabled clients to avoid using extra Gnutella connections when they already have global connectivity.

On a quantitative level, the charts following the topology maps summarize the Gnutella traffic generated in each experiment. The first set of charts are pie-charts which show the relative amounts of bytes that were exchanged between local and global neighbors. In each of the configurations except the smallest, using our connectivity logic reduced the ratio of global to local traffic significantly.

The final chart reports the total amount of bytes exchanged between local and global neighbors in each experiment as well as the overall total number of bytes sent between neighbors. One immediately notices that the total amount of traffic between neighbors in the experiments where our connectivity logic was employed is greater than in the corresponding control. Ignoring the trivial first configuration, a further inspection of the data reveals one of our main results: *our connectivity logic always decreased the amount of global traffic in the network*. In the most dramatic case, the eighty-one node configuration, the total amount of bytes sent between neighbors doubled when our locality logic was enabled, but the total number of bytes exchanged between global neighbors decreased by nearly two-thirds! Putting congestion matters aside, and given our economic assumptions, ISPs would greatly benefit from our connectivity logic.

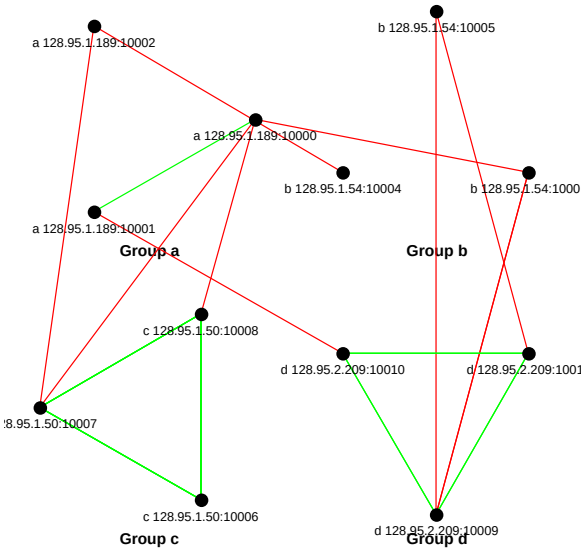

Figure 4: 4 groups. 3 clients/group. 4 connections/client. No locality logic.

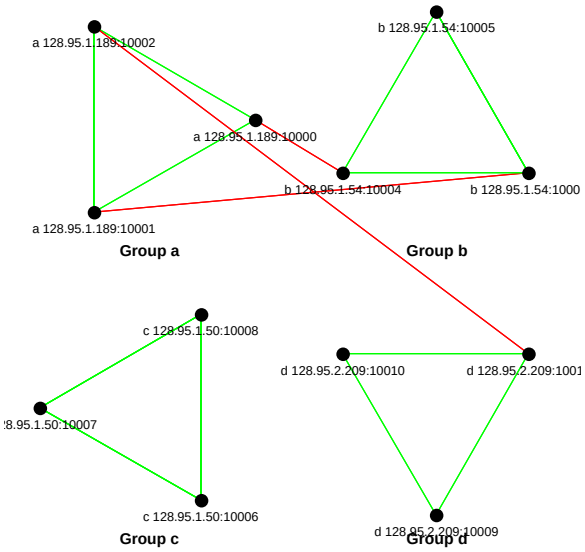

Figure 5: 4 groups. 3 clients/group. 4 connections/client. Locality logic.

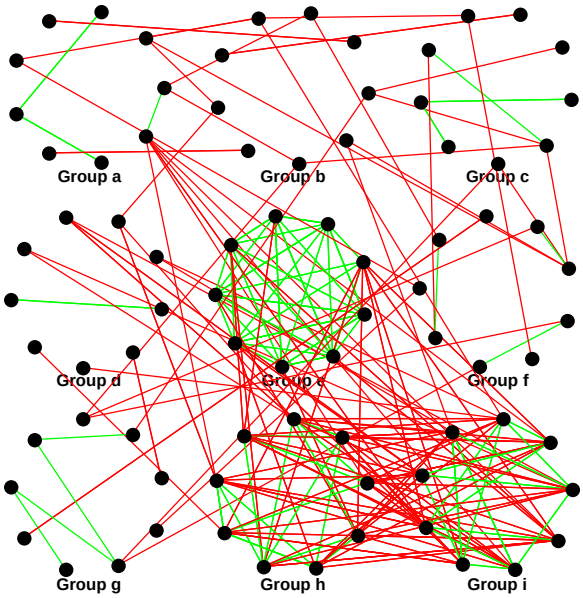

Figure 6: 9 groups. 9 clients/group. 11 connections/client. No locality logic.

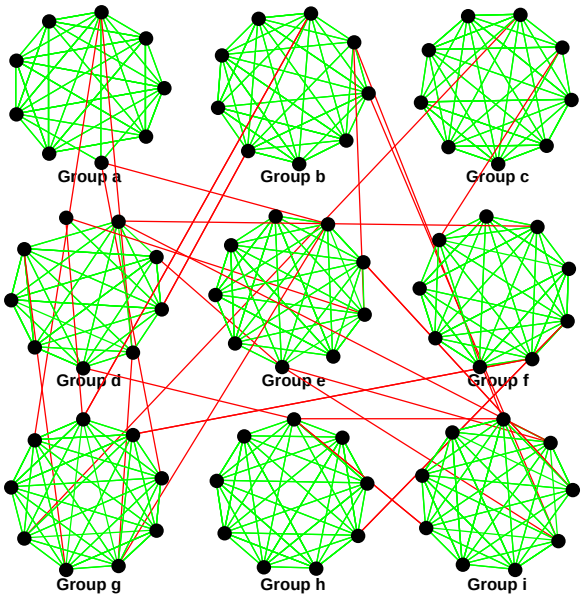

Figure 7: 9 groups. 9 clients/group. 11 connections/client. Locality logic.

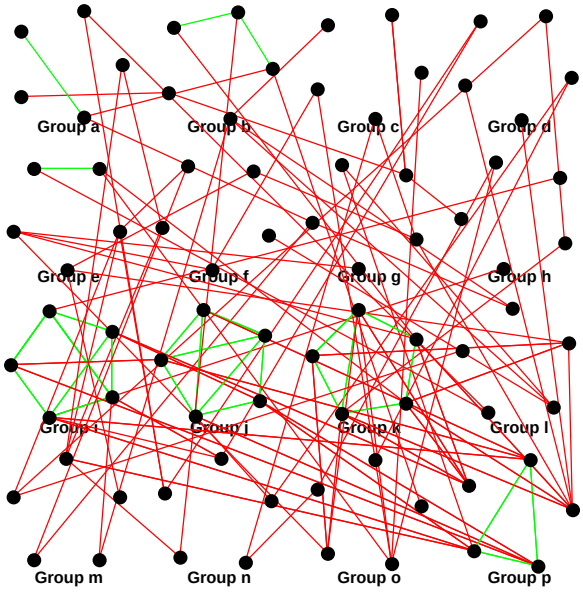

Figure 8: 16 groups. 5 clients/group. 5 connections/client. No locality logic.

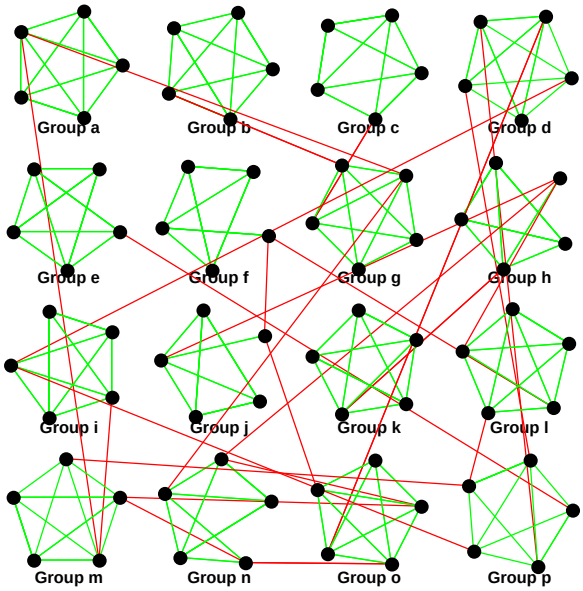

Figure 9: 16 groups. 5 clients/group. 5 connections/client. Locality logic.

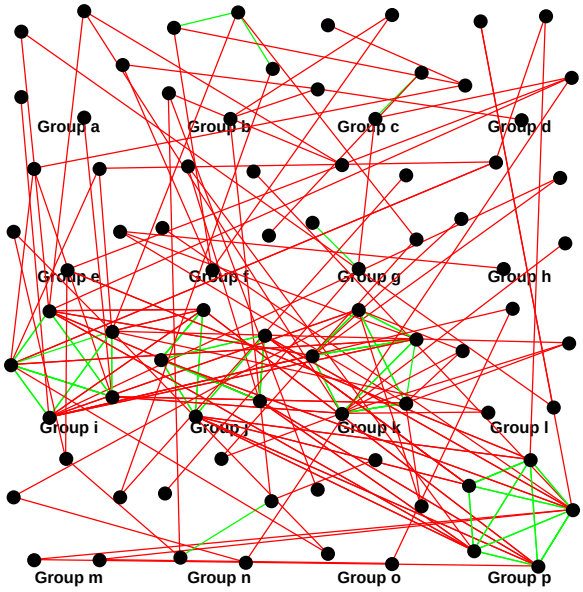

Figure 10: 16 groups. 5 clients/group. 7 connections/client. No locality logic.

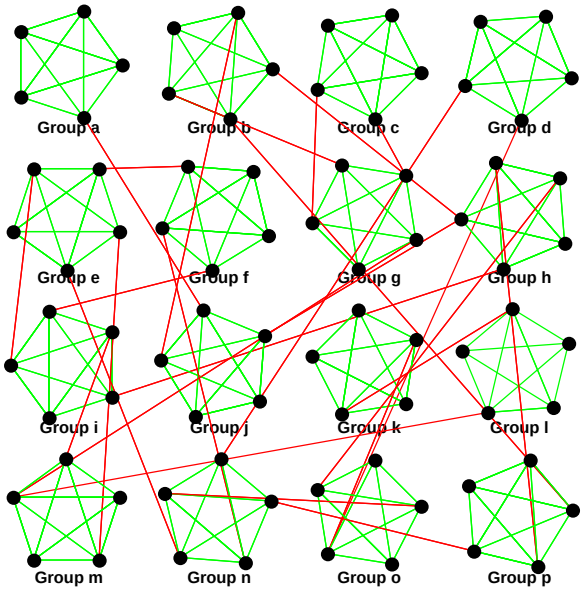

Figure 11: 16 groups. 5 clients/group. 7 connections/client. Locality logic.

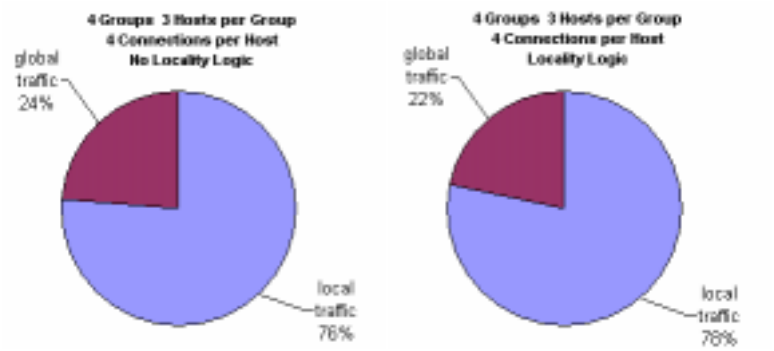

Figure 12: Breakdown of local/global traffic. 4 groups. 3 clients/group. 4 connections/client.

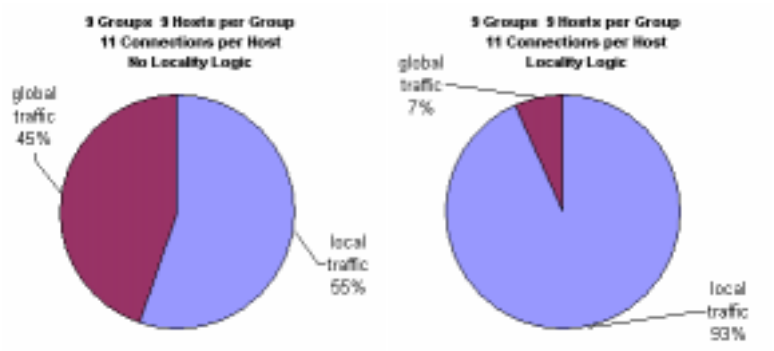

Figure 13: Breakdown of local/global traffic. 9 groups. 9 clients/group. 11 connections/client.

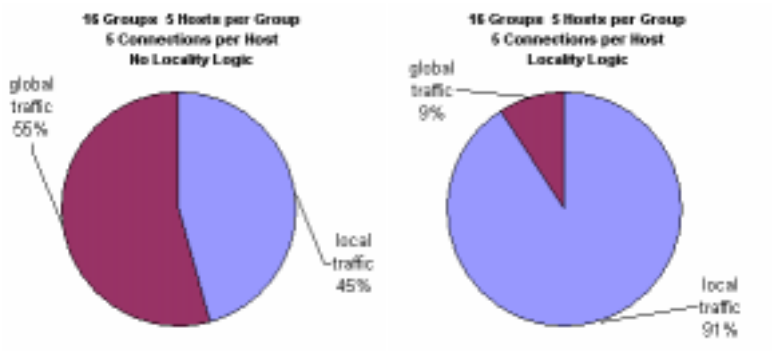

Figure 14: Breakdown of local/global traffic. 16 groups. 5 clients/group. 5 connections/client.

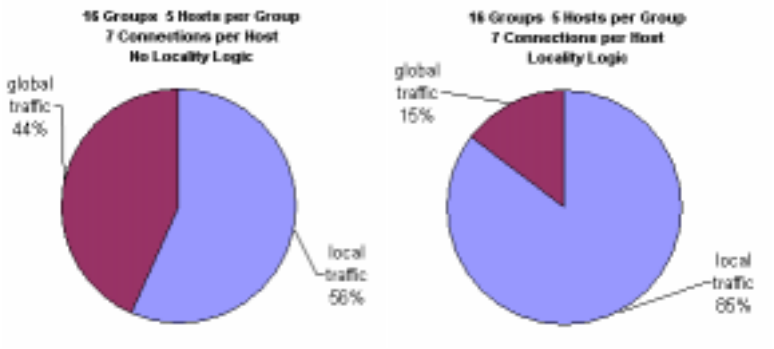

Figure 15: Breakdown of local/global traffic. 16 groups. 5 clients/group. 7 connections/client.

groups	clients/group	cnctn's/client	logic	local	global	total
4	3	4	Y	106411	29962	136373
4	3	4	N	36357	11338	47695
9	9	11	Y	3873683	272407	4146090
9	9	11	N	1118991	918194	2037185
16	5	5	Y	1163726	116304	1280030
16	5	5	N	329049	394468	723517
16	5	7	Y	1357693	234184	1591877
16	5	7	N	358739	276990	635729

Figure 16: Bytes of traffic over (costly) global and (costless) local links, broken down by experimental configuration. Mean of 10 experiments.

7 Future Work

Currently our Gnutella clients only exchange Ping and Pong messages in our experiments. We would also like to simulate query and download traffic. This would be non-trivial, as one would need to decide where to place content and how to generate queries representative of typical user queries found in real peer-to-peer networks. One option for generating query traffic is to have our Gnutella clients randomly choose files to query for or download from a global list of all files in the network. Another option is to dynamically query the network during experiments. A third option is to connect a Gnutella client to the actual Gnutella network, log a trace of the queries received by the client and use this data as a representation of the type of queries present on the Gnutella network in simulations.

Additionally, our technique relies heavily on monitoring Pong and Query messages for an indication that a neighbor is receiving traffic from a non-local client. Pong message volume is directly related to the subtleties of how clients forward and route Ping messages. Dampening the amount of Ping traffic forwarded, perhaps by clients ignoring previously received Ping messages, could lessen overall Gnutella traffic while at the same time hampering the effort to learn the global connectivity state of one's neighbors. We would like to explore the manner in which the various subtleties surrounding descriptor routing affect the topology of the Gnutella network.

Another issue is that pairs of clients in our network can become doubly-connected. That is, client A might have both an incoming and outgoing connection to client B and vice versa. The nature of the Gnutella handshake makes it difficult to iron this bug out of UWtella. In the case that double connections exist, one of the connections is a waste and can prohibit clients from attaining global connectivity. Thankfully, the behavior is correlated to the number of clients in the network: as the number of clients increases, the probability of two clients attempting to connect to one another at the same moment decreases. Thus the smaller tests were most affected by double connections. Redesigning our clients with extra logic to prevent double connections could improve the results for smaller networks, though of course the actual Gnutella network is much larger than any network simulated for this study.

An aspect of Gnutella networks that also deserves attention with respect to cost-effectiveness is file transfers between hosts in the network. Once a Gnutella user decides to download content from another Gnutella user, the transfer occurs over a HTTP connection that is outside of the Gnutella network. Thus if content is available both from a local neighbor and a global neighbor and the user chooses to download the content from the global neighbor, then the ISP must pay. We would like to investigate methods of creating incentives for users, faced with such a decision, to download the content from the local neighbor. Possible incentives include the promise of higher bandwidth or lower latency for

traffic between local neighbors, or only allowing users to download files available from global neighbors when all local neighbors lack the file requested.

Finally, we note that our technique is somewhat independent of the specifics of the Gnutella protocol. We think it could be applied to other self-organizing overlays. We would like to study how other networks could benefit from locality knowledge.

References

[1] Adar, Eytan and Huberman, Bernardo A. "Free Riding on Gnutella" First Monday, volume 5, number 10. October 2000. *http://www.firstmonday.dk/issues/issue5_10/adar*

[2] Clarke I. et. al. "Freenet: A Distributed Anonymous Information Storage and Retrieval System." ICSI Anonymity Workshop 2000.

[3] Friederich, Steven. "Bandwidth restrictions save almost $1 million." The Daily, University of Washington 22 Oct. 2002.

[4] The Gnutella Protocol Specification v0.4. *http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf*

[5] Prinkey, Micheal. "An Efficient Scheme for Query Processing on Peer-to-Peer Networks." Aeolus Research Inc. *http://aeolusres.homestead.com/files/index.html*

[6] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. "A scalable content-addressable network." *Proceedings of ACM SIGCOMM* August 2001.

[7] Ripeanu, Matei. "Peer-to-peer Architecture Case Study: Gnutella Network." *Proceedings of International Conference on Peer-to-peer Computing.* August. 2001.

[8] Rohrs, Christopher and Falco, Vincent. "Improving Gnutella's Ping/Pong Scheme." *http://www.limewire.com/index.jsp/pingpong*

[9] Saroiu, Stefan, Gummadi, Krishna P. and Gribble, Steven D. "A Measurement Study of Peer-to-Peer File Sharing Systems." *Proceedings of the Multimedia Computing and Networking (MMCN)* San Jose. January 2002.

[10] Saroiu, Stefan, Gummadi, Krishna P., Gribble, Steven and Levy, Henry M. "An Analysis of Internet Content Delivery Systems." *Proceedings of the 5th Symposium on Operating Systems Design and Implementation.* December 2002.

[11] Stoica I. et. al. "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications." *Proceedings of ACM SIGCOMM* August 2001