The Coolest Title in the World

Steven Gerding, Jeremy Stribling
{sgerding, strib}@csail.mit.edu

Abstract

This is all very abstract.

1 Introduction

The performance of structured peer-to-peer overlay networks under real-world topology conditions is a relatively unexplored area of research. Even fewer studies exist that compare the relative performance of several different peer-to-peer algorithms and examine their various performance tradeoffs. While recent research has focused on the performance of different peer-to-peer geometries on specific metrics (i.e. static resilience and local convergence) [2], or on examining high-level comparisons between the institution-specific implementations of a few overlays [11], there are several aspects of peer-to-peer performance that have not been addressed by this work.

For example, most peer-to-peer systems can expect a high degree of *churn* in its membership; that is, the rate at which nodes join and leave the network is likely to be high [3, 12]. In order for an overlay to remain connected under such conditions, each node must periodically probe for the liveness of its neighbors. The rate at which each node probes its neighbors incurs a cost in terms of bandwidth, proportional to the amount of state kept by the node. As bandwidth is often the limiting resource in an overlay, it is important to understand the tradeoffs overlays can make to minimize this maintenance bandwidth, while still maintaining reasonable performance.

Another issue is how different peer-to-peer algorithms react to pathological conditions that arise in real-world networks. The impacts of conditions such as asymmetric and time-varying latencies, link failures, and non-transitive links on overlays is poorly understood at best, although we observe that these are all characteristics of a real-world distributed system testbed, PlanetLab [1, 10]. We believe that analyzing the ability of peer-to-peer algorithms to cope with such conditions, and understanding the tradeoffs of this ability on performance, will enable more robust overlay designs in the future.

In this paper, we explore and compare the impact of churn and other pathological network conditions on several different structured peer-to-peer overlay networks: Chord [13], Tapestry [14], Kademlia [9], Kelips [4], and (maybe) Koorde [5]. These overlays vary in their geometries and in the amount of state maintained, leading to differences in performance/efficiency tradeoffs. Furthermore, each algorithm has many different parameters that can be finely-tuned to improve performance and efficiency; we analyze the effect of this knob-turning for each overlay and explore the parameter space to find the best performance envelope for different network and churn conditions. We also compare the different overlays to each other, and discuss the tradeoffs a system designer can make when choosing, configuring, or designing an overlay. We quantify this performance/efficiency tradeoff by comparing bandwidth consumed per node (in bytes per second) to the average latency of a lookup (in milliseconds); this allows us to correctly account for background maintenance traffic as well as timeouts incurred during lookups due to stale routing data.

We perform our evaluations on p2psim, a peer-to-peer simulator recently developed by the Parallel and Distributed Operating Systems group at MIT. The topologies we use for our simulations are extracted directly from latency and failure data gathered on PlanetLab [1]; another contribution of this paper is the presentation and analysis of several different features of this data. Using real-world data in a simulator enables us to compare a non-trivial number of overlay networks under conditions that could actually arise in an actual deployment situation over long periods of time, without requiring the multi-institution collaboration, massive debugging effort, and several months of testing that would be necessary to gather the same data on PlanetLab itself. There are already several overlays and overlay services running continuously on PlanetLab, but often the set of participating nodes is hand-tuned by researchers to avoid nodes that exhibit strange network behavior. One goal of our work is to understand the effects this behavior has on overlays, enabling researchers to design systems that can easily run under a wide variety of network conditions.

The next section provides an overview of previous work in peer-to-peer overlay performance analysis and comparison, comparing and contrasting it to our approach.

2 Related Work

These are all the people who we're cooler than: [2], [11], and maybe [6]. Maybe talk about other overlays (Pastry, Gia, Skipnet, Coral, etc?). Talk about why we didn't use those other protocols (time, similarity, etc.)

Ben the TA wants us to talk about some other papers

[7,8].

3 Data Set

Here we will explain the APP data set, and its various characteristics. We should also talk about why it's appropriate for our experiments [1]. I'm not sure if we want it organized in subsections like this, but for now why not. Give general stats for the static network case, like mean, median, 5th and 95th percentiles. Probbaly show a CDF.

*** Talk about why PlanetLab is cool, and a hint as to why we didn't use other possible data sets (RON, King, Gnutella/Kazaa/Overnet traces) [nontransitivity, time-varying, length]. More detail than intro. Talk about why services deployed on PlanetLab would be important. Continuously running services (Chord, Bamboo, PIER, NYU stuff?, web proxy?). ***

3.1 Asymmetry

Sometimes the round trip times between two nodes is not symmetric. Quantify and graph. I can imagine presenting this data either as just a number (i.e., the number of links that vary asymmetrically by more than 10% or something) or maybe a full graph, like a CDF showing the percentages by unique pairs of nodes.

3.2 Time-Varying

Moreover, round trip times vary over time. Quantify and graph. Maybe this could be an intensity-style graph, where we have all pairs of nodes on one axis, time on the other axis, and let white be the minimum rtt between them, and black be the maximum. Another way would just be to graph it for random pairs, like we have now.

3.3 Non-Transitivity

A can get to B, B can get to C, but A can't get to C. This is a problem, especially with iterative routing. Quantify and graph. Maybe this could just be a single number, like the number or percentage of non-transitive triplets, as well as an explanation/quantification of the inet1 and inet2 only nodes.

3.4 Failures

Talk about link vs. node failures. Give basic churn rates. Maybe talk about half-life here if deemed appropriate by some higher force. Quantify and graph. Maybe a CDF of MTTR and MTTF like in Brent's paper, although if we only use a month's worth of data this might not be interesting. Could also just give a graph of number of nodes alive over time (using the 99% rule or something, modulo non-transitivity), as well as the number of paths available over time.

4 Algorithms

Now a brief summary of all the overlays we tested, and what parameters we varied for each one. I think we can steal most of this from the IPTPS submission.

4.1 Chord

Chord is simple and awesome [13].

4.2 Tapestry

Tapestry is way complicated [14].

4.3 Kademlia

Kademlia is also complicated, but has this cool distance metric (XOR) and this other cool stabilization trickery [9].

4.4 Kelips

Kelips is not $\log n$ performance, but a two-hop scheme. It's crazy [4].

4.5 Koorde

We might be able to get some Koorde numbers, but I'm not sure. Anyway, there are some cool de Bruijn properties here [5].

5 Evaluation

Here's the meat. Lots o' graphs, with the different DHTs in different network scenarios. Included a detail analysis of each graph, saying what it means for that particular DHT, and for DHTs in general. These subsections are just going to be a guide, we'll have to decide soon very specifically what tests we'd like to run. This ordering is dubious, as well. Need different workloads too.

As an alternative, we could organize this by protocol instead by test type, but that seems weird.

5.1 Experimental Setup

Talk about the simulator, failure penalties (if any), no bandwidth/queue modeling, number of nodes, etc.

5.2 Static

Just plain lookups. Serves as a baseline performance metric.

5.3 Churn

Just plain churn, both lookup intensive and churn intensive. Make sure this is cohesive, because this is a node properties while the other tests are link properties.

5.4 Non-Transitivity

Look, this breaks some protocols but not others. Woopdee-doo. We're so awesome.

5.5 Time-Varying

Will this be interesting at all?

5.6 Link Failures

This might be very similar to Section 5.4, maybe we can combine them. May actually end up ditching this altogether if we can't come up with a good story for handling packet loss/link failures for each protocol in simulation.

5.7 Full Gambit (aka Full Fury)

Try all protocols on *all* pathological toplogy conditions, see which set of parameters is best for each protocol. If different from the above cases, make recommendation for self-tuning parameters.

6 Discussion

Here we sum everything up, and talk about our findings across tests and protocols. Make broad observations justified by data, that system designers can use to make useful decisions.

7 Conclusions

I imagine this will be a very important section; we'll need to convince people that they learned something new from our paper. This should not be the type of conclusion where we just rehash the intro, it needs to say something intelligent.

Acknowledgments

Thomer, Jinyang, Robert, Frans, Hari + TAs? Anyone who reads it and gives advice. Maybe Brent Chun for graph ideas.

References

[1] The PlanetLab All-Pairs-Pings Project. http://pdos.lcs.mit.edu/∼strib/pl_app/.

[2] Gummadi, K., Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S., and Stoica, I. The impact of DHT routing geometry on resilience and proximity. In *Proceedings of SIGCOMM* (August 2003), ACM.

[3] Gummadi, K. P., Dunn, R. J., Saroiu, S., Gribble, S. D., Levy, H. M., and J.Zahorjan. Measurement, modeling, and analysis of a peer-to-peer fi le-sharing workload. In *Proc. of ACM SOSP* (Oct. 2003).

[4] Gupta, I., Birman, K., Linga, P., Demers, A., and van Renesse, R. Kelips: Building an effi cient and stable P2P DHT through increased memory and background overhead. In *Second IPTPS* (2003).

[5] Kaashoek, F., and Karger, D. Koorde: A simple degree-optimal hash table. In *Second IPTPS* (2003).

[6] Liben-Nowell, D., Balakrishnan, H., and Karger, D. R. Analysis of the evolution of peer-to-peer systems. In *2002 ACM Symposium on Principles of Distributed Computing* (Aug. 2002).

[7] Loguinov, D., Kumar, A., Rai, V., and Ganesh, S. Graph-theoretic analysis of structured peer-to-peer systems: Routing distances and fault resilience. In *Proceedings of SIGCOMM* (August 2003), ACM.

[8] Manku, G. S. Routing networks for distributed hash tables. In *2003 ACM Symposium on Principles of Distributed Computing* (2003).

[9] Maymounkov, P., and Mazieres, D. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proc. of IPTPS* (2002).

[10] Peterson, L., Anderson, T., Culler, D., and Roscoe, T. A blueprint for introducing disruptive technology into the internet. In *HotNets-I* (2002).

[11] Rhea, S., Roscoe, T., and Kubiatowicz, J. DHTs need application-driven benchmarks. In *Proc. of IPTPS* (2003).

[12] Saroiu, S., Gummadi, K. P., Dunn, R. J., Gribble, S. D., and Levy, H. M. An analysis of internet content delivery systems. In *Proc. of OSDI* (Dec. 2002).

[13] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. Chord: A scalable peer-to-peer lookup protocol for internet applicati ons. *IEEE/ACM Transactions on Networking* (2002), 149–160.

[14] Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., and Kubiatowicz, J. D. Tapestry: A global-scale overlay for rapid service deployment. Special Issue on Service Overlay Networks, to appear.