Quality of Service Routing in Peer-to-Peer Overlays[*]

[Extended Abstract]

Michael Gellman
Dept. of Electrical & Electronic Eng.
Imperial College London
m.gellman@imperial.ac.uk

1. INTRODUCTION

Peer-to-Peer (P2P) overlays have been used to support a number of different applications: from their origins supporting file-sharing, they have expanded to encompass ever more real-time and interactive applications such as streaming multimedia, Voice-over-IP, and real-time gaming. Each of these applications requires different degrees of Quality of Service (QoS); for instance, a file-transfer application requires a path with the highest available bandwidth, while an interactive, real-time application will have latency and jitter requirements. However, the current approach of many P2P overlays is to establish direct connections between overlay participants using the underlying IP routing mechanisms This disregards the potential for using the overlay to exert control over the path that an application's packets take through the network.

One of the most promising approaches to overcoming the limitations of Internet routing is an overlay network. These have shown the benefits in terms of increased QoS of adding even a single network hop in today's Internet. RON [1] was the first to be implemented in a wide-area network; however, it did not scale beyond approximately 50 routers due to the costly $O(n^2)$ overhead associated with probing each overlay participant. When considering only the QoS metric of availability, [3] showed that a simple one-hop source routing scheme could mitigate 56% of the network failures in their study. However, none of these existing overlays have considered the problem of scaling towards an Internet-sized population such as represented by today's P2P networks. Our work hopes to combine the ability of a routing overlay to improve performance within a peer-to-peer framework for scalability.

We do this by leveraging existing work in QoS routing in wired networks [2] as a starting point. We use a *reinforcement learning*-based approach to learn the best route between two nodes in a completely distributed manner. This approach, referred to as the Cognitive Packet Network (CPN) has been shown to be able to autonomously learn the best route in the network in terms of a multitude of QoS objectives; it has been used to find optimal paths in terms of available power, loss and delay, delay and jitter, and path length. CPN uses what it calls *Smart Packets* (SPs) to probe the quality of network paths, communicate these new routes to the source, and at the same time deposit data at the intermediate routers which future SPs use to route themselves.

CPN is particularly attractive because it compactly represents the routing table options as the weights in a neural network, which the reinforcement learning algorithm updates according to the state updates it receives. Also, as we describe below, it can utilize data traffic as probes to gather up-to-date network snapshots; the overhead of the approach is greatly reduced compared to an overlay like RON which uses explicit probe packets, and an aggressive probing strategy to monitor each overlay participant. The network overhead of CPN is limited to an additional header attached to data packets, and the Acknowledgement packets which communicate network snapshots taken by SPs back to the overlay routers.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CoNEXT 2006 Lisboa, Portugal
Copyright 2006 ACM 1-59593-456-1/06/0012 ...$5.00.

2. IMPLEMENTING QOS ROUTING IN A P2P OVERLAY

Previously, CPN has been implemented in wired and wireless networks; the shift to overlay networking presents unique challenges that must be overcome. First, and foremost, is the scaling issue, which we approach by exploiting the same properties of P2P overlays that allows them to scale. Secondly, we must also deal with the question of what to do when the overlay introduces more overhead than it can compensate for with a better path. Finally, we deal with the question of how to send user data using the overlay.

2.1 Scaling the QoS Support

In a physical network, neighbors are those peers who are directly reachable via a given network interface, whereas in an overlay context, *everyone* is connected to each other via the underlying network[1]. However, this problem is neatly solved by taking into account the fact that existing P2P networks (such as Gnutella) have been shown to scale to millions of nodes. In [5], Gnutella was shown to scale through two key features: its gossip-based self-organizing overlay construction protocol, and its hierarchical structure. Its gossip protocol efficiently disseminates neighbor information,

[1]We assume that all peers are *virtually connected* to each other via the underlying network. We omit nodes who are not directly addressable (*i.e.* those behind Network Address Translation firewalls).

allowing new overlay peers to quickly discover other overlay participants, and their status. In addition, it divides overlay participants into two classes, peers and ultra-peers. This exploits the fact that some overlay peers are stronger than others, relying on the former to perform more work than the latter. We exploit this classification system by relying on ultra-peers to perform routing for peers, and excluding peers from any routing duties, thus reducing the number of available routing decisions at any one overlay peer.

2.2 Learning to use the Overlay

Another important factor is that, because the underlying network provides a functional route, we cannot make the assumption that the overlay *must* be used. Rather, we will *learn* when it is appropriate to use the overlay to improve performance, and when traffic will be better off using the underlay. We accomplish this by adding an extra decision at each router in the network (including the source). If this choice is selected as a Smart Packet's next hop, it can be interpreted as the decision to go directly to the destination (*i.e.* the overlay egress). Thus, the decision to bypass the overlay entirely is made at the source using *exactly the same process* as deciding a next hop. The quality of the underlay is compactly represented at each router in the network, and no special treatment for the decision to exit the overlay need be given.

2.3 Handling User Data

Finally we note that, depending on the QoS type, SPs may or may not carry user data. The consideration here is that, while there is no danger of the SP being lost as is the case in a wired environment, an application which is affected by out-of-order deliveries (*e.g.* TCP) may wish to not have its packets encapsulated in SPs, while other applications (*e.g.* UDP) may be less sensitive, and may even benefit since their data packets will use optimal routes as they are discovered. In addition, by using data packets to function as probes, we reduce the total number of new packets created by the overlay, reducing overhead. Our preliminary experience with the overlay implementation has confirmed that UDP benefits from SP encapsulation, and we will explore its impact on the TCP-based applications in the future.

2.4 Implementation

We have initially implemented this algorithm using the Click modular router [4] in C++. It runs as a user-level process, with multiple configuration files. One core file specifies all of the overlay-specific operations common to all applications (*e.g.* sending Smart and Dumb packets, storing routes, and performing the reinforcement learning operations), which is then combined with smaller configurations which contain any of the ingress and egress-specific operations like capturing an application's packet, possibly changing the source and destination address, and injecting it into the CPN overlay.

Currently, we require that both the ingress and egress nodes run the overlay software, which is attractive because it allows us to capture QoS data on the end-to-end underlay path for our decision making.

3. EVALUATION

For reasons of space, we do not provide a full set of results from our implementation. Rather, we summarize two experiments using our current overlay implementation that we have conducted using 46 Linux PCs, configured using topology data of an ISP backbone, complete with artificial delays on each link.

The first experiment took place in a network where the underlying routing protocol (OSPF) had already constructed the set of optimal routes using delay as the cost[2]. Our key conclusion was that we were able to verify that our overlay could learn to use this route, as opposed to its own, which would invariably increase the delay by using unnecessary hops. We also found that the overlay introduces minimal overhead.

Our second experiment's goal was to measure the overlay's ability to route around a *hotspot* in the underlying network. To create the hotspot, we increased the artificial delay on one link. Because the underlying OSPF costs do not change, normally routed traffic will suffer increased delays. We have been able to show that the CPN overlay is able to quickly detect the disturbance along the optimal route, and discover a new route with better performance.

4. CONCLUSIONS AND FUTURE WORK

We have presented a method for enabling QoS-based routing in a P2P overlay. This consists of 3 extensions: using ultra-peers as routers, compactly representing the choice to exit the overlay, and allowing for the option to encapsulate user data in Smart Packets. Based on the experiments that we have conducted, we have shown that this routing algorithm can autonomously learn to use the underlay when no better route exists, and also route around decreases in QoS using the overlay.

The next step for this work is to compare it to an existing overlay such as RON, expand its scope to consider overlay dynamics, and test its performance "in the wild". We have combined our existing routing overlay framework with an existing open-source Gnutella peer-to-peer client, and will begin deployment in the near future. By using the global PlanetLab testbed for a prolonged evaluation, we plan to demonstrate our approach as a scalable approach to QoS routing in P2P networks.

5. REFERENCES

[1] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. *SIGOPS Oper. Syst. Rev.*, 35(5):131–145, 2001.

[2] E. Gelenbe, R. Lent, and Z. Xu. Measurement and performance of a cognitive packet network. *Journal of Computer Networks*, 37(6):691–701, December 2001.

[3] K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall. Improving the reliability of internet paths with one-hop source routing. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation*, 2004.

[4] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, 2000.

[5] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing unstructured overlay topologies in modern P2P file-sharing systems. In *Internet Measurement Conference*, 2005.

[2]This will converge to the optimal set of routes because the OSPF cost is directly proportional to the artificial delay.