Peer Assisted Content Distribution over Router Assisted Overlay Multicast

George Xylomenos, Konstantinos Katsaros and Vasileios P. Kemerlis
xgeorge@aueb.gr, ntinos@aueb.gr, vpk@cs.aueb.gr
Mobile Multimedia Laboratory & Department of Informatics
Athens University of Economics and Business, Patision 76, Athens 104 34, Greece

***Abstract*— While multicasting is considered valuable for content distribution, it is not widely supported on the Internet, despite the emergence of scalable overlay schemes. Content providers have instead turned to peer assisted content distribution in order to efficiently serve large numbers of clients, thus removing the bandwidth bottleneck from their side but placing a heavy burden on the clients. Even if we assume that multicast will become prevalent in the future Internet, peer assisted content distribution will still be useful for asynchronously distributing very large amounts of data. We have thus designed a multicast variant of BitTorrent, paying special attention to the incentives required to ensure that peers will not only consume, but also contribute content. To provide a multicast substrate for our application in the current Internet, we also present an overlay multicast scheme inspired by Scribe that exploits co-operative access routers so as to improve the distribution trees.**

I. INTRODUCTION

As the Internet is evolving from a network connecting pairs of end hosts to a substrate for information dissemination, something clearly indicated by the increasing traffic loads due to peer assisted file distribution, it seems that the Internet architecture should itself evolve accordingly. The *Publish-Subscribe Internet Routing Paradigm* (PSIRP) project [1] is working on redesigning the Internet based on publish-subscribe principles throughout the protocol stack. In this model publishers announce available data, subscribers express their interests, and the network allows them to rendez vous for the exchange of data. To realize this paradigm, we need both applications that operate in a publish-subscribe manner, as well as network mechanisms for rendez vous and data distribution.

While the PSIRP project is still designing solutions for the rendez vous and data distribution needs of a publish-subscribe network, a choice already committed to is the reliance on multicast as the main method of data delivery. Even though multicast may seem to make content distribution trivial, the peer assisted approach in the form of the popular BitTorrent application [2], remains useful for distributing very large amounts of data: BitTorrent allows peers to download parts of the content, leave the system, and then return to proceed from where they left off, unlike in regular multicast where the sender and the receivers need to be synchronized in time. We have therefore designed a multicast based peer assisted content distribution application, based on BitTorrent. To test this application in the current Internet, we use the Scribe overlay multicast scheme [3], modified to exploit access router assistance. In this paper we discuss the design of both the content distribution application and the multicast routing scheme.

Our work was supported by the ICT PSIRP project under contract ICT-2007-216173.

II. THE BITTORRENT APPLICATION

While the protocol used between BitTorrent clients is a (de facto) standard [2], many application details, such as the peer selection strategy, are left to the implementation, so our description below is generic. When a content provider wants to distribute a data set, it organizes it as a sequence of bytes, splits the sequence into equal size *pieces* (e.g. 256 Kbytes or 4 Mbytes) and calculates the checksum of each piece. A *tracker* is then located, that is, a server willing to assist the file exchange. Finally, the tracker address, piece size, total size and all checksums are recorded in a *metafile* which is distributed over the Internet.

After a client locates and downloads a metafile, probably via a search engine, it becomes a BitTorrent *peer* by querying the indicated tracker for a list of peers currently participating in the content's distribution; these hosts comprise what is known as a *swarm*. A client constructs a bitmap with the pieces that it already has, initially empty for new clients and full for the content provider. Then, the client chooses some peers based on any criterion it likes, and attempts to exchange bitmaps with them. Based on these bitmaps and data such as path delays or bandwidths, the client selects peers to exchange pieces with and the pieces to request from them. Pieces are exchanged on a tit-for-tat basis, but to bootstrap new clients, peers give out some pieces for free. Clients normally prefer those peers that provide them with the best service, occasionally contacting new ones in order to discover whether better options have become available.

Breaking down the content distribution into pieces has important implications. First, as discussed above, the exchange becomes asynchronous, allowing peers to exchange the pieces they need independently of the content provider. Second, only the metafile content needs to be trusted: each peer can independently verify downloaded pieces via the checksums and, since pieces are exchanged on a tit-for-tat basis, peers serving bad or no content will receive bad service. Third, hotspots are avoided if enough peers exist in the swarm: each client can select nearby and/or unloaded peers to exchange data with.

While extremely popular, BitTorrent does have some performance limitations. First, the tracker becomes a bottleneck when many peers exist: as it can only return a limited number of peers to each requesting client, the client may not even be aware of the best choices available. Note that the trackerless mode of BitTorrent simply allows some of the peers to operate as trackers themselves, therefore it does not solve this problem. Second, the exchange can be very inefficient: many nearby nodes may be downloading the same pieces from a faraway node, since they make their choices independently [4]. Third, peer selection is expensive: a peer may no longer be available (left the swarm), unwilling to reply (too many connections), not useful (no pieces to exchange) or not satisfactory (low bandwidth). Essentially, each client spends a lot of resources to heuristically search for good peers that host the required pieces among those peers returned by the tracker. A proposed remedy is for access routers to advise their clients on the quality of the network paths towards candidate peers, so as to help them avoid problematic paths [5]; clients however still need to contact all other peers.

III. Applying multicast to BitTorrent

As the most important properties of BitTorrent, that is, support for asynchronous content distribution, lack of trusted third parties and avoidance of hotspots, all arise out of the key decision to break down the content into pieces, in the multicast variant of BitTorrent we maintain this choice by *distributing each piece over a separate multicast group*, thus maintaining the decentralized nature of BitTorrent. A group identifier can be generated for each piece by hashing either the metafile name and number of the piece, or its checksum. At first, a *rendez vous* (RV) point will be located and a distribution tree will be built for each group via an overlay multicast scheme, as explained in Section IV, but in the future these tasks will be performed by the publish-subscribe network, treating each piece as a separate publication, with no changes to the application.

In multicast BitTorrent, a client simply joins the multicast groups for all or some of the pieces that it is missing and then waits for data to start arriving. After a piece has been received correctly, the client leaves the tree. When a client wants to send a piece however, it must first ensure that some receivers do exist for it, so as to avoid wasting resources. To achieve this, the multicast scheme should be able to indicate whether a group is empty or not, or, equivalently, whether a multicast tree currently exists for the group. The RV point should also maintain a minimum interval between replies to sender queries so as to spread piece transmissions in time, thus preventing duplicate transmissions and allowing more receivers to join the group before a piece is delivered.

The sender may chose among non empty groups based on any criterion it likes, for example, the length and/or bandwidth of the path towards the RV point for the group. The sender should also include its bitmap along with any piece transmitted, thus enabling all receivers to identify trees that are likely non empty without spending resources to query RV points. To avoid synchronization, each receiver that has some of these pieces should choose a random one for transmission. Finally, starvation can be avoided if each sender periodically queries the RV points for all the pieces that it has.

For a peer assisted content distribution scheme to operate well, it must provide the incentives for good peer behavior. In BitTorrent this means that peers should not only receive, but also send pieces. When peers directly exchange pieces on a tit-for-tat basis, a client that sends invalid or no pieces is punished by its peers and has to rely solely on random free piece offers. When multicast is introduced, the sender may not even be aware of the, possibly numerous, receivers that it is serving. As a result, while each receiver can detect if a sender is sending invalid pieces via the checksums, it cannot punish that sender since it does not know which peers are receiving the pieces that it is itself sending.

In order to reuse the tit-for-tat approach over multicast we need to ensure that whenever a sender transmits a piece, each receiver will multicast one of the pieces that the sender is interested in. One way to achieve this is for the sender to encrypt some bits in a piece, before sending it along with the bitmap expressing its needs to the group. In order to decrypt the piece, each receiver will have to transmit, also partially encrypted, one of the pieces requested by the sender. At this point the original sender will unicast the decryption key for its piece to each compliant peer, expecting to get in return their own decryption keys. If a key is not returned or is useless, the corresponding peer is blacklisted.

To reduce overhead, the sender only needs to encrypt a part of the piece that is large enough to thwart attempts by the peer to guess its content and verify its guess via the checksum. If the checksum is an n bit hash of the piece, by encrypting $n+k$ bits of a piece, 2^k out of the 2^{n+k} possible combinations would match the hash value. For example, with 160 bit SHA-1 hashes and 256 encrypted bits, an exhaustive search of the 2^{256} possible combinations would provide 2^{96} matches to the hash value. The cost of the extra unicast transmission for the key can be made negligible by choosing a reasonably large piece size.

IV. Router assisted overlay multicast

Since neither IP multicast nor the publish-subscribe architecture of PSIRP are available to us, we need an alternative multicast facility to test our peer assisted content distribution approach. One option is to employ an *Application Layer Multicast* (ALM) scheme [6], where multicast is simulated by unicast transmissions between group members. In ALM schemes however, each member of the group needs to be aware of most, or even all, other members in order to achieve good routing performance, meaning that these solutions are not scalable.

A more scalable solution is to create multicast trees over a *Distributed Hash Table* (DHT) substrate such as Pastry [7], where a large identifier space is distributed among nodes, which co-operate to route data tagged with an identifier to the node assigned with that part of the identifier space. Scribe [3] achieves multicast distribution over a DHT substrate by mapping each group to an identifier and making the node

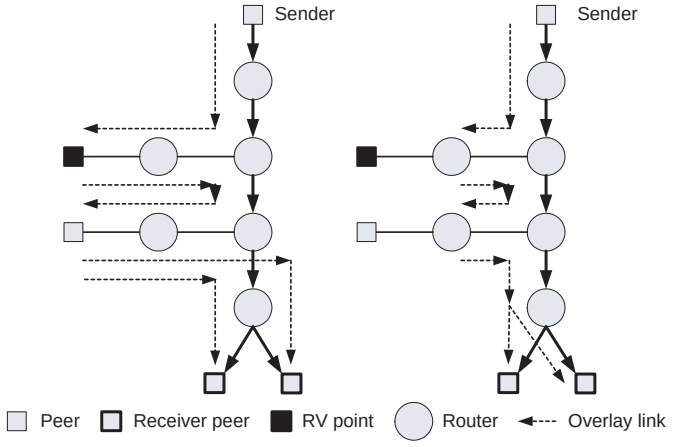

Fig. 1. Overlay multicast (a) without router assistance and (b) with router assistance.

responsible for that identifier the RV point of the group. Receivers join the group by sending a join message towards the RV point; as the message propagates towards it, reverse path routing state is established until a node already in the tree is found, thus forming a multicast tree rooted at the RV point. A sender simply routes data towards the RV point, which then propagates it over the tree. Multicast BitTorrent can operate over Scribe by having each peer in a swarm join the DHT, mapping each piece to an identifier, and then using this identifier to route either join messages (for receivers) or query and data messages (for senders) towards the RV point. The RV point can detect if its group is non empty by checking its Scribe routing state.

Normally DHT nodes are end hosts that use the underlying IP transport transparently to the routers. This however means that a host that is an interior node in many trees will limit their available bandwidth to that of its access link. To avoid this, we can exploit the properties of the underlying DHT to create a set of trees such that each node will be an interior node for only one of them [8]. While this seems ideal for multicast BitTorrent which employs a different tree for each piece, it is tied to a specific overlay routing scheme (in this case, Pastry). In addition, an end host that is an interior node even for a single tree may still be a bottleneck: as shown in Figure 1(a), data in transit has to enter and exit such nodes via their access links; if the access links are asymmetric, the tree bandwidth will be limited by the, typically lower, uplink bandwidth.

To avoid this problem, we are exploring the option of using the access router of a peer as its *proxy* in the DHT substrate and overlay multicast scheme. In this case, as shown in Figure 1(b), data do not need to cross the access links of interior tree nodes, only crossing the, typically faster, downlink direction of access links leading to group members. In Figure 1 this means that a piece transmitted to the group will only cross 10 instead of 14 links, avoiding the uplinks. As a result, the distribution trees become shorter and redundant transmissions are prevented. Preliminary simulation results indicate that while Scribe, running on top of Pastry, produces trees comprised of sender to receiver paths with delays of at most 3 times those of IP multicast, our approach reduces this factor to 2.

Since overlay multicast is a response to the lack of router support for IP multicast, proposing that access routers should implement a DHT substrate and an overlay multicast scheme seems counter intuitive. However, while in IP multicast *all* routers must participate, in our scheme access routers participation is *optional*: the scheme can operate without router assistance, albeit with reduced efficiency. In addition, while many routers have no incentive to participate in IP multicast routing, in our scheme access routers acting as proxies for their attached end hosts will, first, reduce their traffic load by eliminating transmissions over their access links and router-to-router links and, second, provide an enhanced service to their customers, as the end hosts will experience lower latencies and higher bandwidths. These are the same arguments that motivated network providers to offer Web and other application proxies to their clients.

V. Summary and ongoing work

We have presented the design for a peer assisted content distribution application that maintains the basic architecture of BitTorrent but modifies the piece exchange process to operate over multicast, as well as for a router assisted variant of the Scribe overlay multicast scheme that leads to more efficient distribution trees. Both proposals fit into the publish-subscribe Internet framework of the PSIRP project, the former as a platform for exploring the application design space, and the latter as a means of running such applications over the Internet. While multicast BitTorrent avoids the costly unicast peer selection process and provides more efficient data distribution, it is important to understand how factors such as piece size and the sender policy for querying the RV points affect its performance. As the preliminary simulation results for our router assisted overlay multicast scheme are encouraging, we are currently implementing multicast BitTorrent in our simulator, assuming a scenario where a very large data set is distributed to numerous users over an Internet like topology.

References

[1] PSIRP Project Team, *PSIRP Project Home Page*, http://www.psirp.org.
[2] BitTorrent development community, *Protocol specification*, http://wiki.theory.org/.
[3] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 100–110, 2002.
[4] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should internet service providers fear peer-assisted content distribution?" in *Proc. of the Internet Measurement Conference*, 2005, pp. 63–76.
[5] H. Xie, Y. Yang, A. Krisnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider portal for applications," in *Proc. of ACM SIGCOMM*, 2008.
[6] Y.-H. Chu, S. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1456–1471, 2002.
[7] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *Proc. of Middleware*, 2001.
[8] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in cooperative environments," in *Proc. of Symposium on Operating Systems Principles*, 2003.