ThriftyTorrent: Cheap P2P software patch distribution for enterprise networks

Miranda Mowbray, Chris Peltz, Cristiano Costa, Nazareno Andrade, Katia Saikoski

HP Laboratories

HPL-2009-80

Keyword(s):

P2P, security

Abstract:

We present a peer-to-peer method of distributing software patches or other data in an enterprise network, which aims to minimize the cost of bandwidth used subject ta a specified quality of service in the form of a target download rate.

External Posting Date: April 21, 2009 [Fulltext]   Approved for External Publication
Internal Posting Date: April 21, 2009 [Fulltext]

ThriftyTorrent: Cheap P2P software patch distribution for enterprise networks

Miranda Mowbray (HP Labs), Chris Peltz (HP Software), Cristiano Costa (HP Labs), Nazareno Andrade (Federal University of Campina Grande, Brazil), Katia Saikoski (HP Brazil R&D)

Abstract

We present a peer-to-peer method of distributing software patches or other data in an enterprise network, which aims to minimize the cost of bandwidth used subject ta a specified quality of service in the form of a target download rate.

The need for cheap software patch distribution in enterprise networks

There is a commercial trend for consolidation of servers and data centres, in which server and storage resources in enterprise networks are now used for applications throughout the enterprise, instead of only by their local "island" [1]. An island might be one department, or one site, or all the enterprise machines in one country. This trend leads to more efficient use of servers and storage, but more network traffic, and hence higher system costs for use of bandwidth. In particular, a server sending software patches may now be responsible for patching machines separated from the server by links for which the costs of bandwidth use are much higher than was the case before consolidation.

In enterprise networks, the bandwidth of some links (e.g. transatlantic leased lines) may be charged by use, at high rates compared to the cost of traffic on local links. During certain periods of time software patch distribution can account for a large fraction of the total data traffic in large enterprises [2]. We obtained data from one enterprise patch server that served 7,355 machines in a large enterprise network over 66 days: on a peak day it sent a total of over a Terabyte of data in response to requests from five continents, and on an average day it sent over 50 GB of data.

The problem we address is how to distribute software patches from a small number of servers to machines throughout the enterprise (which may be other servers, or desktop PCs that need to be patched) at as low a bandwidth cost as possible, subject to a specified quality of service. The quality of service requirement that we use is a target for the rate at which a requested patch arrives. This problem is relevant both to HP's own network and to enterprise networks managed by HP on behalf of customers, and our solution could also be sold as part of the HP Software portfolio to enable enterprises to reduce the bandwidth cost of patching their own machines. This could include both the Opsware datacenter automation offering and the Configuration Management product line.

Our Solution

To solve this problem we have designed ThriftyTorrent, a peer-to-peer (P2P) protocol for enterprise software patch distribution. The peers are machines in the enterprise network. The network management system informs each peer *x* of a set of neighbor peers of *x*, an estimated cost per megabyte to deliver data from each neighbor peer, and a target download rate for *x*. The management system can trade off control/manageability against optimal data placement by its choices of neighbor sets: *x* will only download from its neighbors. The target rate may be different for different classes of patches: for instance, patches curing security vulnerabilities may be sent urgently. Peer *x* estimates the download rate on the route from itself to each neighbor peer, using bandwidth estimation techniques and its recent experiences of downloading from that peer. The use of recent information makes ThriftyTorrent adaptive to changes in availability of peers or congestion of links.

The patches are partitioned into smaller pieces. When peer *x* downloads a patch, it identifies the set N of neighbor peers possessing pieces of the patch that it does not yet have. It then identifies the subsets of N for which the sum of estimated download rates from peers in the subset to x is greater than or equal to the target download rate. If there is no such subset, peer *x* downloads pieces from all the peers in N. If there is at least one such subset, peer *x* picks such a subset P for which the sum of the costs per megabyte to distribute data to *x* from peers in the subset is smallest, and downloads pieces from the peers in P. See Figure 1.

Since different pieces may be downloaded from different machines, this can create a swarm of peers which upload and download pieces to each other simultaneously: such swarms have been used very successfully by BitTorrent [3] to reduce resource requirements of content providers. In networks for which less detailed information is available on bandwidth costs – for example, Extranets linking several enterprise partners – our solution can make use of what information there is to reduce the use of routes with known high bandwidth costs.

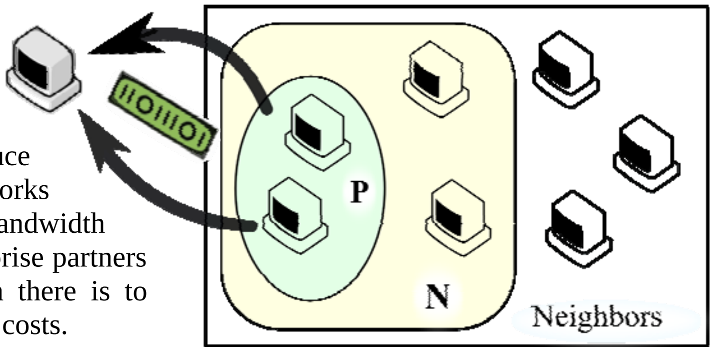

Figure 1: illustration of the ThriftyTorrent protocol

Competitive approaches

Currently the most common way to distribute software patches (and other content) in enterprise networks is to use a client-server architecture with a small number of mirror servers. All machines download the patches directly from one of a few servers, so this makes no use of the bandwidth between clients, and requires the servers to have high performance capabilities. The route between a client that requires a patch and another client that already has it is quicker and cheaper than the routes from the downloading client to any of the servers with the patch.

One way that ISPs attempt to reduce bandwidth costs is to redirect requests for content outside the ISP to local caches or other sources for the content inside the ISP, using traffic manipulation products such as those provided by CacheLogic, Cisco or Sandvine[1], or Content Delivery Networks such as Akamai[2]. Enterprises can use this approach to eliminate costly long-distance transfers of software patches that are already available from a local source, however patches which are not available locally may still use expensive routes to travel across the enterprise network. Moreover, these solutions require caches or Content Delivery Networks with high performance capabilities in terms of bandwidth and robustness: these can be expensive to build. Using a P2P system to deliver patches, as ThriftyTorrent does, has the advantage that it does not require traffic manipulation, robust high-bandwidth caches, or Content Delivery Networks: it can be implemented using lightweight clients at the peers. Moreover P2P delivery can adapt more flexibly and quickly to changes in connectivity and availability than may be possible for competitive approaches relying on a fixed network element at each location.

Almost all of the prior art on P2P delivery of software patches (eg. [2] and US patents 5835911, 7162538, 7178144, 6950847) ignores bandwidth costs. An exception is the suggestion by Waldvogel and Rinaldi [4] and by Liu et al. [5] of a P2P network in which a peer chooses where to download from by selecting a location with the cheapest bandwidth to the peer. The problem with this approach is that the cheapest routes may be slow. In contrast, ThriftyTorrent takes speed of delivery into account as well as cost.

Evidence that our solution works

We have used trace-based simulations to evaluate the performance of ThriftyTorrent. For input data we used a month's HTML logs from a Linux Common Operating Environment (COE) patch server on the a large multinational enterprise network, recording some 700,000 successful requests for files. Our simulation ignored bandwidth contention in internal links and effects from overlapping of requests. We used a model of the network (based on real data about its core network) in which local connections were fast and cheap compared to long-distance connections on the core network. To make a fair comparison we set the target download rates for ThriftyTorrent equal to the download rates obtained using client-server distribution.

We compared ThriftyTorrent to client-server distribution, and to the P2P distribution method suggested by [4,5] which minimizes bandwidth costs. Figure 2 shows the cumulative bandwidth costs in the simulations of the three methods, and Figure 3 the sum of the distribution times in the simulations. The values plotted in both Figures are normalized to make the value for client-server distribution reach 1 at the end of the month.

[1] http://www.sandvine.com/general/getfile.asp?FILEID=16
[2] http://www.akamai.com/html/solutions/media_delivery.html

Using ThriftyTorrent in place of client-server distribution saved a third of the bandwidth cost. Using the cheapest possible P2P distribution only saved 45% of the bandwidth cost. ThriftyTorrent's average delivery speed was 20% faster than the average delivery speed for the other two methods, which had approximately the same average delivery speed. Moreover 99.97% of the patches were delivered at least as quickly by ThriftyTorrent as by client-server distribution.

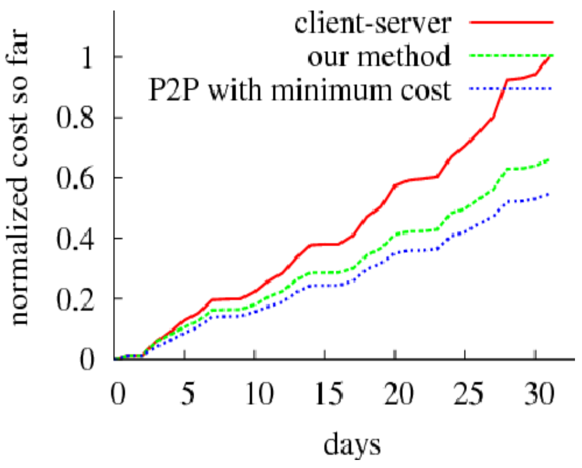

Figure 2: cumulative bandwidth cost for distribution

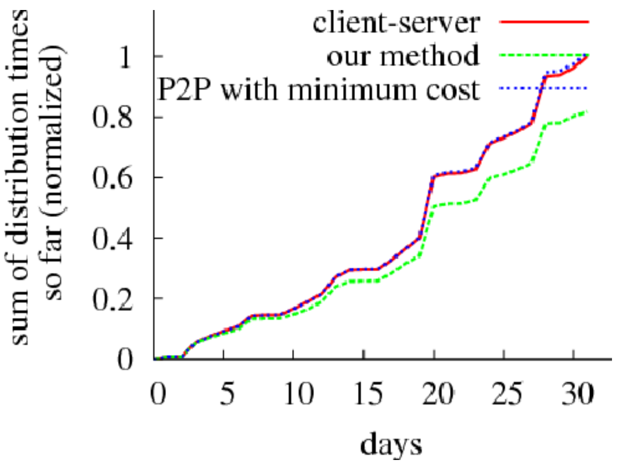

Figure 3: cumulative distribution time

Next steps

So far we have concentrated on patch distribution. ThriftyTorrent is also potentially applicable to the distribution of media files within an enterprise, the distribution of disk images for operating system provision, or the deployment of software applications.

A possible next step is to determine the importance and impact of our approach to HP customers looking for datacenter automation and/or software provisioning solutions, perhaps using their logs or audit records as simulation data, and to refine the approach based on this customer feedback. This validation would investigate both cost improvements and the overhead on clients, and would include a Proof of Concept using HP Software products to evaluate effectiveness beyond just a simulation.

References

[1] Hewlett-Packard Development Company, "Getting off the high-cost technology island", HP e-newsletter, http://h71028.www7.hp.com/enewsletter/cache/412974-0-0-224-121.html, 2007
[2] C. Gkantsidis, T. Karagiannis and M. Vojnovic, "Planet scale software updates", ACM SIGCOMM Computer Communication Review 36(4), pp. 423-434, October 2006
[3] B. Cohen, "Incentives build robustness in BitTorrent", Workshop on Economics of Peer-to-Peer Systems, June 5-6, 2003
[4] M Waldvogel and R Rinaldi, 2003, "Efficient Topology-Aware Overlay Network", ACM Computer Communication Review 33(1), pp.101-106, January 2003
[5] Y Liu, Z Zhuang, L Xiao and L M Ni, "AOTO: Adaptive Overlay Topology Optimization in unstructured P2P systems", Proc. IEEEE Global Telecommunications Conference vol. 7, pp.4186-4190, 2003