

The Virtual Multicasting Approach to Bandwidth Conservation

Brynn Andrew and Philip Machanick

*Department of Computer Science, University of the Witwatersrand
{bandrew,philip}@cs.wits.ac.za*

Abstract—Virtual Multicast (VMC) exploits short-term similarities in Internet traffic to save bandwidth and reduce congestion. Its implementation requires modification of aspects the HTTP specification, which is motivated by the expected benefits of the method: reclamation of bandwidth lost due to redundant network data and latency savings as seen by users.

I. INTRODUCTION

Internet congestion is a growing problem: as capacity increases, so does demand. During peak times, this congestion manifests itself as a phenomenon that can bring down web-sites and networks. Given that there could be a significant common traffic at peak times, it is reasonable to investigate sharing common data as far as possible.

Information Mass Transit (IMT) is a general design philosophy aimed at exploiting commonality of data on a medium to reduce bandwidth demands and improve latency [1]. By analogy with mass transit, as opposed to private transport, IMT aims to reduce congestion by exploiting commonality in traffic patterns.

Virtual Multicasting (VMC), as a specific instance of IMT, is a method that attempts to exploit short term similarities in Internet traffic. VMC combines identical data-streams on a network, resulting in a better model of content delivery.

This paper defines the problem which VMC attempts to address and the way in which it addresses the problem.

This research aims to show that by changing the current model of content delivery to one which has more awareness of the content being delivered, i.e. with VMC, congestion will be eased and latency reduced.

Some preliminary measurement has been done on FTP logs to justify the approach, though initial work as described in this paper will focus on modifications to the hypertext transfer protocol (HTTP).

The remainder of the paper is structured as follows: Section 2 defines the conditions under which VMC

is most effective. Section 3 outlines the protocol changes necessary for the implementation of VMC. Section 4 describes a test implementation of VMC. Preliminary results supporting the concept are given in Section 5. Finally, conclusions and future work are presented in Section 6.

II. INTERNET CONGESTION

General congestion on the Internet is problematic. It manifests itself in slow downloads, intermittent connections and irritated users. At times though, congestion can become so bad that the computers hosting web-sites crash and surrounding networks become congested to the extent that they are almost impassable. This research, although addressing general congestion, focuses on this catastrophic type of congestion.

Congestion is caused by a number of factors. Firstly, the physical media can be a limiting factor – it is not possible for a 10Mb/s LAN to provide the service of a 100Mb/s LAN without upgrading the entire network. Furthermore, network design is limited by nature. It is not possible to move data at a rate faster than the speed of light. The latencies and delays introduced by the physical network thus have to be endured.

Secondly, studies have shown that most of the data downloaded comes from a small percentage of available Internet sites [2]. This implies that the same data is being transmitted in multiple sessions across interconnected networks – a phenomenon most noticeable when a new version of software is released. Millions of people may hit the site almost simultaneously, causing congestion in the servers and surrounding networks.

Linked to this is the current method of content delivery. At present, users wanting access to a World Wide Web (Web) site make a single point-to-point connection to the site in order to retrieve the information. This connection cannot be shared with other users wanting access to the same site, even if the two users are sitting next to one another.

Finally, most Internet bandwidth is owned by large Internet Service Providers (ISPs) or infra-structural carriers (such as large telecommunication compa-

nies), so most Internet traffic is concentrated over their networks. A tree-like network hierarchy, with the larger ISPs at the top of the tree, is created as a result. The topmost nodes of these networks have data requests from all over the world flowing through them.

All these factors combine to cause congestion.

At times of heavy usage, however, the situation can become one where the same data is being downloaded from the same Internet sites from all around the world. Since much of this data is being transported over the same networks (over the high-level nodes of large ISPs), a large percentage of data being downloaded on the Internet is simply being multiplexed in time. It is under these circumstances that congestion can increase to such an extent that servers and their surrounding networks fail. It is also under these circumstances that VMC aims to be most effective.

If a popular way to access the Internet is to download the same content as many others are downloading, why not design for this case, rather than the current situation, where this usage pattern is in some respects a worst-case scenario? This question leads to the idea of VMC.

III. VIRTUAL MULTICASTING

This section outlines the idea behind VMC, and illustrates the general idea with an example. The example is kept relatively simple to illustrate the principles.

A. VMC Concepts

Virtual Multicasting is a method that changes the current model of content delivery to one that is more aware of the connections being maintained over the network. By being more aware of the connections, redundant data can be removed from the network, which in turn can decrease bandwidth usage, reduce congestion, and improve latency for users.

VMC is intended to be implemented as an extension of IP routing, in which common TCP streams are identified, and combined. VMC exploits very short-term data locality, and locality across a large part of the Internet (traffic through the busiest routers).

VMC differs from standard multicasting [3] in that there is no explicit setup of multicast routing trees and data is delivered reliably. The name multicasting is used because both virtual multicasting and multicasting share the trait that only a single stream is used for the same data as far as possible. VMC is different from caching in that it occurs at a high level of the hierarchy (highest-traffic segments and routers), rather

than the endpoints.

In a sense, VMC is a combination of caching and multicasting. From caching, it uses the idea of accessing data from a source closer to the client than the origin server [4]. From multicasting it takes the idea of dissemination of data to many users via a single stream [5]. The combination aims to produce a method that combines the best of caching and multicasting without many of their problems [4, 6].

The key principle behind VMC is that during times of heavy usage, some of the information traveling over links on the Internet is going to be identical [7]. Virtual Multicasting attempts to exploit this to reduce congestion and speed up data access.

For example, when a popular new piece of software has been released (such as a new version of a web browser), many people will want to download it at similar times. As the first person accesses the site, a TCP connection is setup between their client machine and the server. In the traditional model of content delivery, every person connecting to the server has a separate TCP connection. In a simplified version of the Virtual Multicasting model, further accesses to the same data will be served by the original stream – i.e. users will get the data *multicast*¹ to them transparently.

Once the original stream has finished, a request is made by the client to fill in any gaps in the data.

B. Example

To illustrate the principles, here is a simple example.

A simple network is shown in figure 1. Clients 1, 2 and 3 all want to download the same data from the server.

This network abstracts the real-world situation: the router represents high-speed connections at the top of the hierarchy, and the connections to the server and clients could go through multiple routers. The connection to the router on the client side in figure 1 represents the point where routes to the clients diverge.

The following sequence of steps relates to the abstracted example. In a real-world version, there would be other routers, but those closer to the server would not see the requests that are serviced by VMC.

Step 1: Client 1 issues a request for the data (a Web page or FTP file). The router passes the information to the server and references data stream by the data request.

¹The use of multicast in this case refers to multiple streams of data being collapsed into a single stream of data. It is not the same multicast mechanism as that developed by Deering and Cheriton [3].

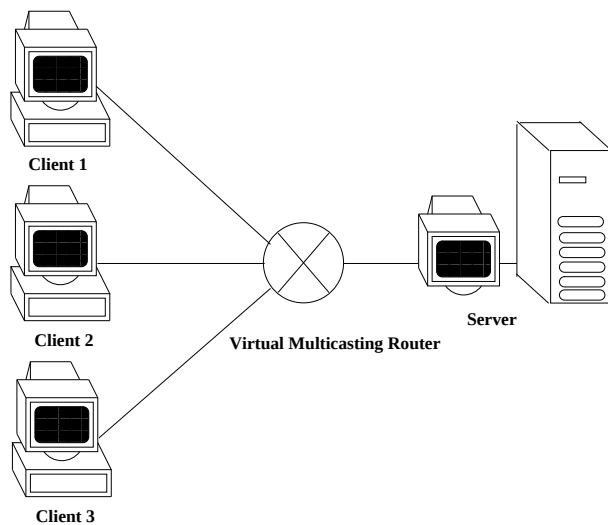


Figure 1: Example Network

Step 2: A few seconds later, Client 2 requests the same information. The router intercepts this request, recognising it as a request for the same data, copies the data from the current stream to Client 1 and sends it to Client 2.

Step 3: A few seconds later, Client 3 issues a request for the data. As in Step 2, a new stream with the same data is sent to Client 3. Client 3 is going to be receiving data at a later stage in the transmission though.

Step 4: The data transmission ends. At this stage, Client 1 has all of the download, Client 2 has $n\%$ of the download and Client 3 has $m\%$ of the download (where $n > m$). Clients 2 and 3 then make a request for the parts of the data they are missing.

By making the clients request the data they are missing, the overhead is reduced on the router. Furthermore, making the client issue further requests fits in with the HTTP model [8].

Ideally, VMC should be completely transparent. However, in our initial work, we are prepared to make simple modifications to standard protocols to demonstrate feasibility. The modifications that will be made are detailed and motivated below in section IV..

IV. VMC IMPLEMENTATION

VMC will initially be implemented to support HTTP, which is a very popular protocol for the dissemination of data on the Internet. Furthermore, the design of HTTP makes additions to the protocol, without affecting its normal operation, comparatively easy. Other protocols will be investigated once the feasibility of VMC has been demonstrated with HTTP. This section explains some of the changes to HTTP/1.1

that are necessary for the implementation of VMC. Thereafter, an implementation for testing the feasibility of the method will be described.

A. Hypertext Transfer Protocol Adaptation

HTTP has to be adapted in two ways. First, there has to be an indication that the client can support VMC. Second, the client has to be able to handle a response giving the subset of an object even if the entire object was requested. So, some client support is required for the operation of VMC. The client has to be aware in that if it only downloaded the second half of the object it requested, it will then request the first half.

In order for the client to indicate that it can support VMC, an additional HTTP header has to be created. An addition to the HTTP/1.1 RFC [8] can facilitate this. A request header field of the form `VMC-aware-client: yes` will have to be included to let any VMC aware router know that this client can accept virtually multicast data. Without this request header, the request will be routed normally, facilitating compatibility with normal HTTP.

The second part that requires changing is the response of a range instead of an entire object. This situation occurs when client 1 has already begun downloading the object and client 2 attempts to download the same object a part of the way through. The response to client 2's request will contain a Content-Range entity-header (section 14.6 of [8]). Although the HTTP/1.1 RFC does not specifically disallow a response of a range when a whole object has been requested, it is prudent to include such behaviour in the specification of the VMC extension. It is conceivable that some clients might treat such a response as an error, even though such a response is allowed in the specification, since range requests and responses are commonly used by caches.

Finally, there has to be client support for the re-requesting of dynamic ranges. The client must be aware of which parts of the object it has and request the ranges it does not. Satisfaction of this requirement does not require any change of the HTTP/1.1 specification.

B. VMC Test Implementation

The initial implementation will be used to test the feasibility of the VMC method. Feasibility here takes the form of inter-operability with current protocols, overhead that is introduced by the method and finally any bandwidth and latency savings that VMC creates.

There are two main components to the test are the client and the router. For the implementation, as much complication as possible will be kept off the

router and put onto the client. The rationale behind this is that the router is going to be handling many data streams, while the client is comparatively idle.

The VMC router will be implemented on a Linux box with `ip_forward` enabled for it to act as a router. The redirection capabilities of Netfilter [9] will be used to redirect packets destined for port 80 to the VMC software. VMC routing will work as follows:

The Uniform Resource Identifiers (URI) of the request should be stored as well as the range that it is requesting. The VMC router would then open a socket to the origin server, rewrite the request and begin copying the data from the new socket to a dynamically resizing cyclic buffer. The client would then read from the buffer to get the data.

A further request for the same data would be detected through the URI. The VMC router would then know which buffer the new client must read from and point the client to the buffer. Before this happens, the VMC router would write a response to the client indicating the range that is going to be received. The client would then read the buffer for the data.

Once the first client had all the data, the second client would request the range of bytes it did not have to get the rest of the object.

The client has to be modified so that it can request missing ranges and put the ranges together correctly to form the complete object. Instead of modifying an existing client, a simple perl client will be written to deal with the VMC requirements for this research. In practice a transparent cache could be implemented on the client side to provide this operation.

This implementation will give a good idea of the resources required for a VMC prototype.

Once the experiment has been set up, a number of scenarios can be simulated. The simplest scenario is to have the server in figure 1 act as a highly demanded web-server, having to serve the same data to many clients, simulated by the clients in figure 1.

A more complicated scenario is to view the server in figure 1 as the other side of a high-bandwidth link.

By comparing the results of the router operating with VMC active or inactive, the effectiveness of VMC can be measured. This effectiveness will be in terms of bandwidth and latency savings, computation overhead and resource requirements.

The next section details some preliminary results for the VMC method.

V. PRELIMINARY RESULTS

Preliminary results were obtained by analysing the FTP logs from a commercial ISP. This analysis showed that there is significant overlap of FTP traffic, at least from their site. The overlap of traffic would not occur with VMC since streams would be sharing this data. We can therefore compare the amount of data shipped normally versus the amount of data shipped using VMC. This preliminary works illustrates the potential of VMC; to test the proposed implementation, HTTP logs will be used.

The comparison of the number of bytes shipped can be seen in figure 2. The x -axis represents the different log days.

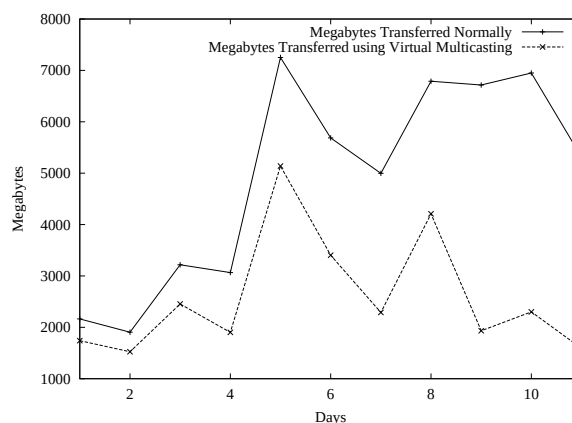


Figure 2: Comparison of Transfer Methods

The graph in figure 3 indicates that the number of bytes shipped using Virtual Multicasting is less than normal data transfer.

A cumulative comparison can be seen in figure 3. Again, the number of bytes are plotted per log-day.

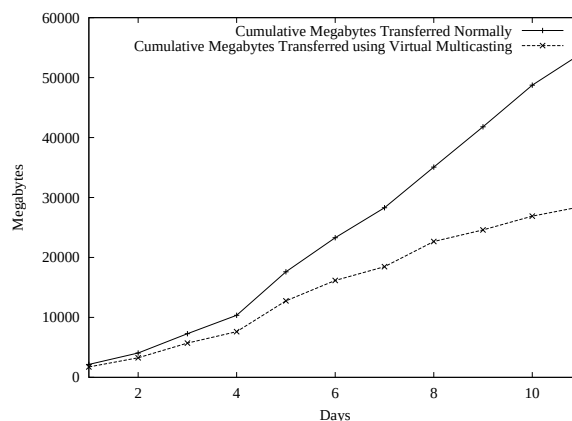


Figure 3: Cumulative Comparison of Transfer Methods

Again, this graph demonstrates the potential for better

performance of Virtual Multicasting.

The total number of bytes transferred normally over the log days examined was 5.67×10^{10} . The number of bytes transferred using the Virtual Multicasting approach was 2.99×10^{10} , 52% less than the normal mode of transfer. The biggest saving through the implementation of Virtual Multicasting was 71% and the smallest was 19%.

These results were obtained by doing very rudimentary calculations (not taking bandwidth and latency issues into account) on the data. This indicates the considerable promise that VMC shows and points to further investigation. Further experimentation will give a more focussed idea of the gains of VMC.

VI. CONCLUSIONS

Virtual Multicasting is an alternative content delivery mechanism and compared with existing mechanisms is more aware of the traffic that is flowing over a given network. Through VMC, congestion on the Internet can be eased and catastrophic congestion can be avoided through the better use of the available bandwidth.

VMC is not difficult to implement and it has been shown in this paper that VMC can exist without significant modification to HTTP.

The preliminary results show that VMC has considerable promise and the planned experimentation should show this in more depth. The experimentation will also show the feasibility of the approach and give a reasonable indication of its benefits and problems.

A. Future Work

Future research work would be to investigate the security implications of VMC. The issue of scalability also has to be addressed. Other future work would be to design and implement a Virtual Multicasting prototype and install it on a live network.

B. Overall Conclusions

Should VMC prove to be the success we anticipate it to be, it will provide a base for further investigations of the information mass transit idea.

Acknowledgments

We would like to thank The Internet Solution for providing logs on which our preliminary study is based, and Oskar Pearson of Qualica for his technical exper-

tise. Scott Hazelhurst helped with the final typesetting.

The NRF provided financial support.

REFERENCES

- [1] P. Machanick, "Streaming vs. latency in information mass-transit," *Computer Architecture News*, Vol. 26, No. 5, pp. 4–6, December 1998.
- [2] A. Rousskov and V. Solokiev, "On Performance of Caching Proxies." WWW Document, August 1998. <http://www.cs.ndsu.nodak.edu/~rousskov/research/cache/squid/profiling/p%apers>.
- [3] S. E. Deering and D. R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Transactions on Computer Systems*, Vol. 8, No. 2, pp. 85–110, February 1990.
- [4] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox, "Caching Proxies: Limitations and Potentials." WWW Document, December 1995. <http://ei.cs.vt.edu/~succeed/WWW4/WWW4.html>.
- [5] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "An Architecture for Wide-Area Multicast Routing," in *Proceedings of SIGCOMM'94*, pp. 126–135, 1994.
- [6] B. Rajagopalan, "Reliability and Scaling Issues in Multicast Routing," in *Proceedings of SIGCOMM'92*, pp. 188–198, 1992.
- [7] B. Bashforth and C. Williamson, "Statistical Multiplexing of Self-Similar Video Streams: Simulation Study and Performance Results," in *Proceedings of the Sixth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 1998.
- [8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP 1.1. RFC 2616." WWW Document, June 1999. <ftp://ftp.is.co.za/rfc/rfc2616.txt>.
- [9] M. Boucher and R. Russell, "The netfilter project: Packet mangling for linux 2.3+." WWW Document, 2000. <http://netfilter.kernelnotes.org>.