

Latency Improvement in Virtual Multicasting

Philip Machanick¹ and Brynn Andrew²

¹ School of ITEE, University of Queensland
Brisbane, Qld 4072, Australia
philip@itee.uq.edu.au

² School of Computer Science, University of the Witwatersrand,
Johannesburg, Private Bag 3, 2050 Wits, South Africa
brynn@cs.wits.ac.za

Abstract. Virtual multicasting (VMC) combines some of the benefits of caching (transparency, dynamic adaptation to workload) and multicasting (reducing duplicated traffic). Virtual multicasting is intended to save bandwidth in cases of high load, resulting from unpredictable but high demands for similar traffic. However, even in cases where relatively low fractions of traffic are similar (hence offering few opportunities for VMC), introducing VMC can have a disproportionate effect on latency reduction because of the generally beneficial effect of reduction in traffic, including reduced contention. This paper presents results of a study of latency reduction across a range of workloads, illustrating the potential for VMC even in situations where the extent of overlapped traffic is light.

1 Introduction

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 [14]. The name derives from an analogy with mass transit, where apparently-slower modes of transport like buses and large passenger aircraft are faster for moving large numbers of people with common destinations than apparently faster alternatives (cars, executive jets). Sharing a common form of transport reduces congestion, and makes better use of common media.

Internet congestion is a growing problem: as capacity increases, so does demand. Given that there could be significant common traffic at peak times, it seems reasonable to investigate sharing common data as far as possible. By analogy with the mass transit idea for moving people, if much traffic at the same time is similar, grouping this similar traffic could have significant advantages.

Virtual Multicasting (VMC), a specific instance of IMT, finds common streams which have started at similar times, and combines them. The general model can vary in different implementations. For example, grouping FTP streams may not introduce significant latency or real-time concerns, provided the streams are large enough that saving transmission time dominates any cost of grouping similar traffic. Where streams can be combined, latency can be reduced, since the server is in effect moved closer. Reduction of congestion (queueing delays, lost or corrupted packets, retransmits because of timeouts, etc.) can also reduce latency.

In this paper, the main focus of the investigation is the effect of VMC on Internet traffic with no special opportunities for VMC. The intent is to show that VMC can offer a significant advantage in reducing congestion, even when the opportunities for reducing overlap are limited.

1.1 Information Mass Transit

A number of applications of the IMT idea have been proposed [14]. The general model is one of sharing a stream; actual realization may vary considerably.

One example is the Scalable Architecture for Video on Demand (SAVoD), which aims to implement a video on demand system which scales up to an unlimited number of users [13]. SAVoD streams multiple instances of a movie continuously, so that a virtual VCR can be implemented by finding a suitable point in any given stream, to perform operations such as fast forward, rewind, or start a new movie. The principle is to invest in a large amount of bandwidth, with the goal of removing all requests to the server. Consequently, the biggest latency problems in scaling up to unlimited users are removed.

VMC is the next attempt at realizing the broader IMT idea.

1.2 Virtual Multicasting

Virtual Multicasting (VMC) aims to exploit short-term similarities in Internet traffic, particularly higher up the bandwidth hierarchy. A high volume of similar traffic may periodically occur as new software is downloaded, a large number of clients join the same audio or video stream, or visit a new web site.

Such traffic cannot easily be cached for two reasons:

- the repeated traffic may be transient, and the demand may no longer exist by the time it is cached
- the users may be widely spaced around the Internet, and only the higher-bandwidth links at the top of the hierarchy may see duplicated traffic, i.e., endpoints are not the right place to cache this kind of traffic

The transient nature of this kind of similar traffic also makes multicasting an inadequate approach to reducing wastage of bandwidth. Setting up a multicast route requires prior knowledge that it is required, which may not be easy to predict, since demand for similar content may be hard to predict in advance.

1.3 Remainder of Paper

The remainder of this paper is structured as follows.

Section 2 provides an overview of the VMC concept and related approaches, as related to the general IMT model. The basis for experiments is described in Section 3, followed by results in Section 4. Finally, conclusions, including possible future work, are presented in Section 5.

2 Background and Related Solutions

Virtual Multicasting attempts to reduce or control Internet congestion. It does this by moving away from the traditional model of content delivery (unicast) to one that makes more effective use of the available bandwidth. Instead of having data distributed from a single point, VMC aims to distribute the dissemination of data, reducing the congestion of servers and interconnected networks, freeing bandwidth and as a result, reducing latency from a user's point of view.

VMC is intended to be implemented as an extension of IP routing, in which common TCP streams are identified, and combined. As opposed to standard multicasting [6], there is no explicit setup, and if a client joins a stream late, it will receive earlier traffic out of sequence, sent as a separate stream.

VMC works by maintaining a record of data travelling on the router. If a new client requests data that the VMC router is transmitting already, the request is not passed to the server. Instead, the router creates a response for the previously-transmitted portion of the data, and copies the current stream to the new client. If the router has previously seen multiple requests for the same content, a new client is simply added to an existing VMC session, and the router can send the missed content to the client from its buffer. The first time a duplication is detected, the router starts buffering content, and has to request the missing initial part of the stream from the original server.

The router ends up with two or more clients receiving the same data from a single source, once the VMC setup is complete.

Once the download is complete for the first client, the clients which joined the VMC session later issue a request for data they missed.

For playing a movie, VMC has potential to reduce latency for viewers by bringing content closer to all but the first recipient. More significantly, reducing congestion will likely reduce latency for all network users, not just participants in the VMC session, given the bandwidth required for a movie. Unlike typical file downloads, a movie can run for more than an hour (2 to 3 hours, if it is a full feature), and relieving load even by finding a single extra viewer could have a significant effect on the network. A movie, however, presents a problem: if the client has missed some initial content, significant buffering would be required at the client side to receive the VMC stream as well as the missing initial content.

Real-time traffic (e.g., Internet radio or TV) should be easier for VMC than other examples, because patching in previous missed content is unnecessary.

VMC can be contrasted not only with multicasting, but also with proxy caches, which save recent content to avoid repeated delivery. VMC differs from caching in that it occurs in the highest-traffic segments and routers, rather than at the endpoints. Further, VMC happens on the fly, whereas caching stores a stream for future use. VMC therefore exploits very short-term locality, and locality across a different part of the Internet.

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 remainder of this section provides a brief overview of conventional multicasting, proxy caches and an experimental VMC implementation.

2.1 Multicasting

IP multicasting is the transmission of a packet to a subset of hosts in a network [7]. It provides packet delivery to these hosts at a lower network and host cost than broadcasting to all hosts or unicasting to each host in the group.

Hosts to whom a multicast is destined share a Class D group address (a class reserved for multicast groups [6]). Routers need to know which hosts are in a group: this can be determined by a router polling hosts, or by hosts informing routers [19]. Multicasting has a high setup overhead: a router needs to construct a spanning tree, pruned to exclude hosts not in a multicast group [4].

Another problem is that many routers on the Internet are not configured to allow the transmission of multicast packets. These routers have to be bypassed by IP tunneling [20], a non-trivial task – as a result multicasting is not widely supported by Internet Service Providers (ISPs).

Multicasting suffers several problems in scaling up, such as the acknowledgement implosion problem, resulting from the fact that many more acknowledgements will be routed back to the sender than the original number of multicast packets [15]. There have been various attempts at addressing the scalability problems of multicasting, including Protocol Independent Multicasting (PIM) [5]. However, PIM introduces yet another standard, which increases the difficulty of providing multicasting capabilities across the Internet.

While there has been some work on using multicasting to support video on demand, the proposed solutions are complex, and still need work [12].

Finally, the “best-effort” attempt at data delivery that multicast operates with, is not good enough for many applications which need data to be reliably transferred. Reliable multicast protocols have been developed, but they are inefficient in the delivery of data and have a propensity to cause packet storms [11].

2.2 Proxy Caching

A proxy cache (often simply called a “cache”) is a service between web servers and clients. Generally, a proxy cache is close to users, and aims to exploit similarities in local demand. It watches requests for web objects (e.g., HTML pages, images and files) and saves a copy of objects locally. Subsequent requests for the same object can then be served from the cache.

A cache is implemented transparently, in that once it is set up, a client need not specifically request content from a cache. The cache intercepts traffic and serves requests it can meet, and passes others on. A browser may be configured to point to a specific cache, but caching can be completely transparent (a client is not configured specifically to use the cache). Caches can reduce latency as seen by clients and reduce the bandwidth used by the clients. Caches can be seen as a congestion avoidance mechanism, since they reduce Internet traffic by storing data locally.

Some incoming data cannot be cached. This is due to factors such as dynamic content and rapidly changing web pages. Studies have shown that the amount of Web traffic that cannot be cached is as high as 20% [18]. Furthermore, even with an infinite cache size, the upper bound for the hit rate is 30-50% [1, 18].

It is not always useful to have a cache hit, because the cache server may be overloaded and unable to serve the object efficiently [17]. Furthermore, the time taken to check the validity of the object might be longer than retrieving the object itself. Caches may also be slower on misses than an uncached connection, since the time taken searching a hierarchy for the object may be longer than retrieving the data from the origin server [18]. Every slowdown in the cache adds to the latency experienced by the user.

Finally, caches are often large, and based on expensive hardware and software which have to be configured and constantly maintained. If there is a problem with the cache server, an entire network may be deprived of Internet connectivity, which may be unacceptable for many applications (e.g. Internet banking).

2.3 Comparison to VMC

The common basis of multicasting and caching is that they are bandwidth saving and congestion reduction mechanisms. VMC uses the single data stream idea of multicasting and the transparent nature of caching to produce a mechanism with the benefits of both, while attempting to limit the costs and problems of multicasting and caching.

Unlike caching, VMC occurs near the top of the hierarchy, so the cost would only be incurred at high-throughput routers, whereas caching occurs at endpoints, and is therefore a highly replicated cost. Caching at endpoints could still catch traffic widely spaced in time, which VMC would miss. Multicasting requires prior knowledge that a stream will be shared, and has a high setup cost. VMC, by focusing on traffic through the highest-traffic routers, reduces the setup cost. Further, the VMC approach of transparently initiating sharing when a need is detected means that it is not necessary to predict the need for sharing in advance. However, where it is known in advance that a multicast session is required, it would still be a viable option where it was supported, since routing could be carried out without the requirement of VMC-aware routers.

Finally, VMC routing is intended to occur only through selected routers near the top of the hierarchy, which means that it is not necessary that a large part of the Internet be aware of VMC routing.

3 Experimental Framework

The main goal of this research is to provide a feasibility study of VMC. It is thus necessary to focus on potential obstacles to VMC's implementation rather than on a complete solution.

While FTP should benefit from VMC, the FTP protocol does not lend itself to simple modification to evaluate our ideas. HTTP has the option of requesting

a range – a feature used by caches [2]. While FTP does have a “restart” option, it is not supported in most file transfer modes [16], which would make sending a missed range of a file more complex than with HTTP. HTTP encapsulates all the file transfer mechanisms of FTP and is widely used as a substitute for FTP. Furthermore, the protocol itself is cleaner and better defined – particularly for our purposes. Our approach therefore was to base our investigation on changes to HTTP to support VMC.

This section presents a brief summary of preliminary results which further justified the research, then outlines an experimental version of VMC. The approach used in experiment described in this paper is described, and, finally, our expectations for results are summarized.

3.1 Preliminary Results

A preliminary study of FTP logs from a commercial Internet service provider showed that there was significant overlap of FTP traffic from their site. The overlap of traffic could be eliminated by VMC, since streams would be sharing this data. We did very rudimentary calculations (not taking congestion and latency issues into account), over 11 consecutive days of logged traffic, of the potential bandwidth savings.

The total number of bytes transferred normally over the log days examined was 5.67×10^{10} . The number of bytes eliminating all overlaps was 2.99×10^{10} , 52% less than the normal mode of transfer. The biggest saving through eliminating overlaps was 71% and the smallest was 19%. This initial study [3] showed that VMC had considerable promise, and was worth further investigation. Clearly, a more realistic experiment was the next step. However, these logs represented a relatively high degree of overlap, so we chose to find other logs where the overlap was much lower, to illustrate the potential for gains across a range of traffic conditions.

Accordingly, our more realistic experiment used logs from another source, with much less overlap.

3.2 Experimental VMC Implementation

Establishing the feasibility of the VMC approach takes a number of forms. First, the actual mechanics of VMC have to be developed and demonstrated. Second, it will be no good if the method exists in a vacuum, so good interaction with the current Internet protocols must be demonstrated. Finally, VMC is likely to add latency. This additional latency must be measured and weighed against latency gains, to decide the effectiveness of the method.

In order to evaluate these feasibility issues, an experimental VMC system has been built. The strategy was to start with a simple implementation, to minimise complexity of understanding the results. Accordingly, a simple network topology was implemented, to abstract the main features of the design. This simplified network implemented a VMC router on a computer with a single web server playing the role of multiple servers. While a real VMC route would be

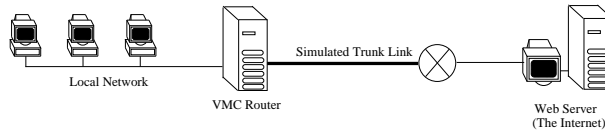


Fig. 1. *Experimental Setup*

several layers away from the servers and client machines, intermediate links were removed to simplify measurement.

The Virtual Multicasting router software was implemented as a simple test bed, designed to experiment with variations on simulated workloads, based on data from cache logs. In the absence of VMC routing, standard IP routing takes place, as a base line from which to compare overheads and advantages of VMC. Fig. 1 illustrates the experimental setup.

The setup is intended to abstract the key requirements of a VMC route: servers providing potentially similar information, and clients with potentially overlapping requirements.

To simulate traffic from a large network, traffic logs from the University of the Witwatersrand cache were used to generate traffic from a single server, with a link approximating the speed of the university's link to the outside world. This traffic had much less overlap than that of our preliminary study.

The VMC router uses the same strategy as a proxy cache for identifying like traffic: it looks for TCP packets with a destination port of 80, and requests are indexed using the MD-5 hash of the universal resource indicator (URI) in the request. A VMC application on the router PC is handed any packets with a destination port of 80, using the `IP_REDIRECT` capabilities of netfilter [10].

Traffic is buffered in the router, and if the same request is detected (by hashing on the URI), it is directed to the buffered content. The VMC router in effect proxies connections, but disguises the fact that it does so from clients by rewriting addresses.

Clients have to be modified so that if they receive partial content, they are aware of this and are able to request the missing data. The router has to send a **Partial Content** response to a client specifying the range of the supplied bytes. Thereafter, the client has to issue a request for the missing range of bytes.

All of these details are contained in the specification of HTTP 1.1 [9]. The only change in usage is that range responses are usually only generated on request. The simplest way of introducing this change would be to add it in to proxy caches, so they would cooperate with VMC routers, but a better long-term change would be to modify the HTTP protocol, so clients could use VMC routers directly. The standard as currently worded does not prohibit clients from dealing with ranges. However, most do not, because a range-response is not a usual outcome of issuing a non-range request, so the proposed change would be to amend the HTTP standard to ensure that clients are implemented to understand a range-response from a non-range request.

However, in this research, we have confined ourselves to evaluating the VMC idea, rather than considering how to change standards to accommodate it.

3.3 Experimental Approach

The experiment reported on here compared a calculated latency gain, based purely on time saved resulting from overlaps in files in a simulated workload, with actual latency gain as measured on a simulated VMC environment. The intent was to evaluate the predictive value of a simplistic measurement, as well as to show the value of even relatively small bandwidth savings in terms of latency improvement. The simulated environment did not take into account latency gains from reducing traffic on multi-hop routes to a client, and therefore underestimates latency gain in a real environment.

The University of the Witwatersrand uses a Squid proxy server to service about 10,000 users. Web requests are logged, and information logged includes that which we needed: time of request, size of the request and time taken to service the request. The size and diversity of the academic community is sufficient to give an approximation to a more general scenario. The phenomenon of self-similarity [8] suggests that our traffic logs are likely to be representative of a wider sample of the real Internet – though the logs we used in our preliminary work suggest that there is a wide variety of traffic patterns.

Our approach was to clean the log files, so extraneous information was removed, as were requests which did not result in data being returned, or which were not well-formed. We then used the logs to generate random bytes up to the length of each request. Had we been exploring issues where the content was significant (e.g., compression), we would not have been able to use random data, but that was not an issue for our experiments.

The data used is selected from real data from 3 days of logs, as well as two artificial pathological cases, representing unrealistically high overlap, and no overlap. The pathological cases are intended to illustrate the extremes: a best-case and a worst-case scenario for VMC. The worst-case scenario provides a measure of the overheads introduced by VMC, since no savings are made (i.e., the only difference is the overhead of trying to find VMC opportunities).

The high-load cases are taken from 4 hours of logs, at busy times of the day, while the low-load cases are taken from 8 hours of logs during quiet times (late at night and early in the morning). The pathological case of no overlap was created by taking a log from a low-traffic period, and eliminating the overlaps. The artificial case of very high overlap with high load was created by interleaving extra requests for a 1Mbyte file as every fifth download.

The calculated latency gain was based on a simple subtraction of the time saved if overlaps identified in the files transferred were removed. The experimental scenarios and calculated latency savings are presented in Table 1. The low-traffic scenarios were generally taken from logs early in the morning on a Monday or late at night on a Friday, when usage was low. The high-traffic scenarios were taken from logs during the day time on a week day, when usage was relatively high. The degree of overlap is relative: as can be seen from the

scenario		Workload	Calculated Latency
load	overlap	files/hour	Saving (%)
low	low	3410.75	0.93
high	high	105322.00	8.12
high	low	898891.75	1.95
low	high	16583.75	2.18
pathological cases			
high	v. high	82221.5	23.05
low	none	3410.75	0.00

Table 1. *Experimental scenarios (calculated latency gains based on examining logs).*

bandwidth saved in the Results section (Table 2), the degree of overlap is not very high except in the contrived case of very high overlap.

The experimentally-determined latency gain was measured as the difference between elapsed time for transmission of the entire workload with and without VMC. This experimentally-determined latency gain is a more realistic measure than the calculated latency gain, since it takes into account the overall effect of VMC on the network, including the extra latency of VMC and improvements resulting from the reduction in network traffic (including reduced congestion).

In our experiments, we eliminated the possibility of high load adding to latency because of limitations of our network cards, by dividing simulation runs (which varied from approximately 25,000 to 420,000 files) into runs of 5,000 files at a time. In a real scenario, this issue would not be a problem because we were simulating activity of an entire campus on a small number of machines.

3.4 Expected Results

Given that the calculated latency savings are only based on reducing the transmission time for the saved bytes, we expected that the measured latency savings would be significantly higher. Any reduction in network traffic will generally improve latency, through reducing collisions (in a network which permits collisions such as ethernet) and generally reducing contention for shared resources.

We expected that the achieved latency gain would therefore be significantly higher than that which was calculated.

Further, we expected latency gains, given the nature of the traffic, to be significantly higher than bandwidth gains. Much traffic resulting from web page access is relatively small files (e.g., an icon, or the text of a web page), which makes the probability of overlaps being significant in size and occurring close enough in time to be useful for VMC to be low. On the other hand, any such overlaps which are found have the potential to reduce congestion. Even in a lightly loaded network, overlaps can potentially lead to short-term hot spots, which VMC has the potential to alleviate.

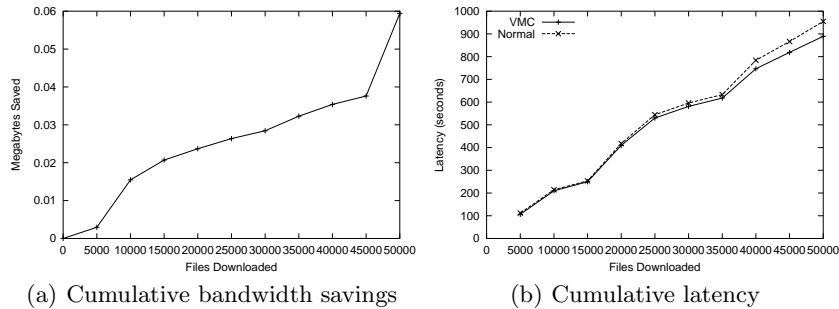


Fig. 2. *Low overlap, low load.*

4 Results

In this section, we present results of experimental bandwidth and latency gains, which are compared with the calculated latency gains. The aim is to highlight the difference between latency and bandwidth gains. VMC is designed to reduce congestion on the Internet, with latency reduction – the measure of most interest to the user – the goal. Accordingly, the focus in presenting the results is on presenting latency reduction as the measure of interest. Bandwidth reduction is also shown as a basis for understanding why latency has improved.

In general, bandwidth savings are modest and on their own do not make a convincing case for VMC. Latency gains, on the other hand, are significant, and do make a case for further investigation of the idea.

The remainder of this section is presented in the following order. First, plots of bandwidth gains and latency variation with and without VMC are shown, followed by a table summarizing results. Finally, the results are discussed.

4.1 Bandwidth and Latency Savings

To illustrate how latency gains can be amplified by hot spots, the latency gains are shown as cumulative plots of latency, compared with plots of bandwidth gains. Total bandwidth is not plotted, because the differences between with and without VMC do not show on any of the graphs, except on the pathological case of very high overlap on a high load.

Fig. 2 illustrates the case of low overlap with low load. As can be expected, total latency saved is relatively small (fig. 2(b)). Bandwidth saved is only 0.02% of the total. However, the overall saving of latency is 6.88%, which compares well with the calculated saving of 0.93% (7.4 times higher). A large fraction of the overlap occurred towards the end of the workload (probably because this workload was measured up to 8am), as can be seen in fig. 2(a).

Fig. 3 illustrates the opposite case: a relatively high load with a relatively high degree of overlap. There are several significant steps in the graph showing saving in bandwidth (fig. 3(a)). These steps correspond roughly to increases

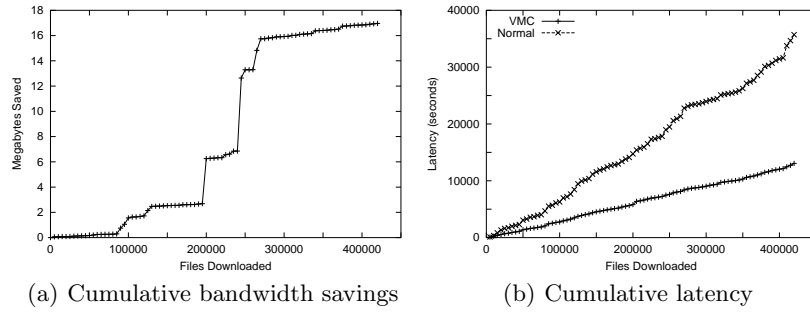


Fig. 3. *High overlap, high load.*

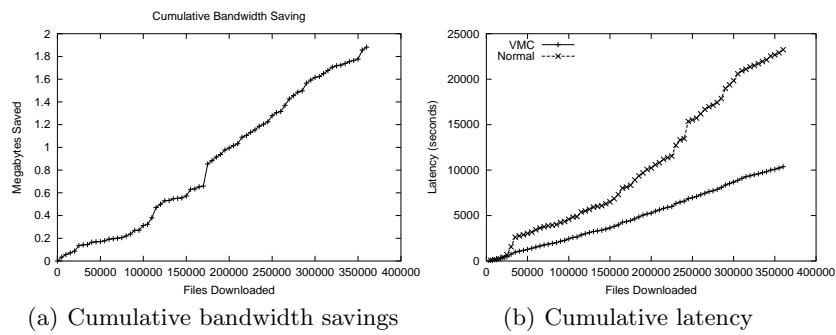


Fig. 4. *Low overlap, high load.*

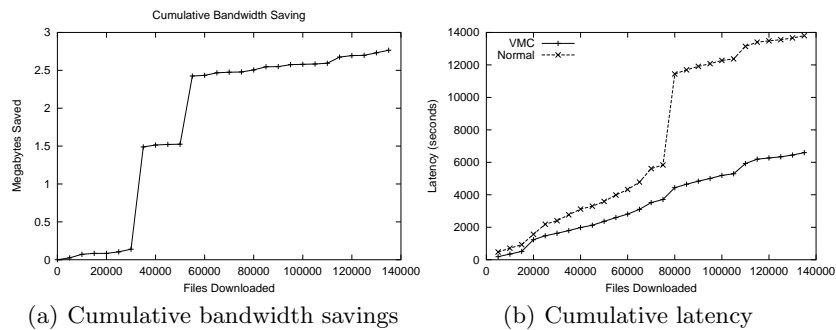


Fig. 5. *High overlap, low load.*

in the bandwidth graphed for the “normal” (non-VMC) case – particularly at about the point where 250000 files have been downloaded. Another observation which is clearer in this case than the low load, low overlap case is that the VMC cumulative latency graph is smoother than the “normal” graph, illustrating the fact that VMC has reduced hot spots.

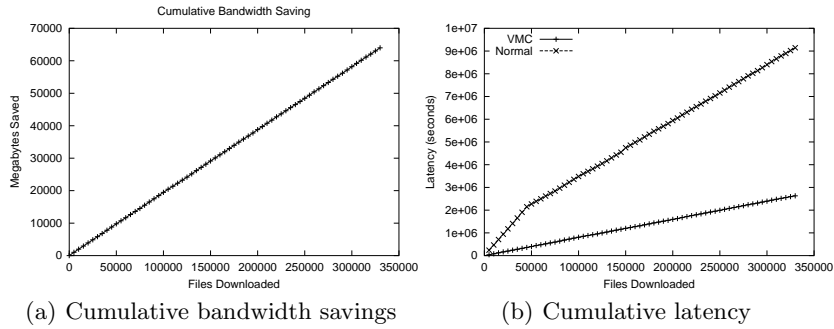


Fig. 6. Artificially high overlap, high load.

scenario		Mbyte transferred		saving %	
load	overlap	Normal	VMC	bandwidth	latency
low	low	338.13	338.07	0.02	6.88
high	high	4263.82	4246.86	0.40	63.42
high	low	3528.81	3526.93	0.05	55.36
low	high	1928.13	1925.36	0.14	52.2
pathological cases					
high	v. high	68556.81	4564.23	93.34	71.21
low	none	381.33	381.33	0	-0.74

Table 2. Measured bandwidth and latency gains.

The case of high load with low overlap (fig. 4) is interesting because it illustrates again how VMC is able to smooth out hot spots, even when they may be relatively uncommon. The overall effect is that, despite only saving 0.05% of the bandwidth, latency is improved by 55.36% overall (as compared with the calculated saving of 1.95%).

The final case of a workload based on a real usage pattern is that of high overlap with a low load, as shown in fig. 5. In this case, again, the value of eliminating hot spots is illustrated. While the total bandwidth saving of 0.14% is very modest, latency overall is reduced by 52.2%. This latency saving is compared with the calculated latency saving of 2.18%. What should be noted specifically here is that the overlaps, while few, are bursty in nature, with each overlap resulting in a big step in the bandwidth savings graph (5(a)).

Finally, the artificially-constructed case of very high overlap (fig. 6) shows how VMC could reduce latency in an extreme case (e.g., a new major software release, a popular movie available for download). In this situation, over 90% of the bandwidth is saved, and the latency improvement is 71.21%, as opposed to the calculated 23.05%. The artificial case of no overlap is not graphed; the result is obvious: graphs for the VMC and “standard” cases are almost identical, except for a small extra overhead on latency for VMC, totalling 0.74%.

Table 2 summarizes the results.

4.2 Summary of Findings

Latency gains, as calculated, varied from 0.93% to 6.69% (excluding pathological cases). These gains translated to measured gains varying from 6.88% to 63.42%. Since these are cumulative measures, they do not convey the improvement which would be seen by a user, where a hot spot in network activity would cause annoying delays. Smoothing out the cumulative latency graphs, as VMC has done in all cases, should translate to a more predictable user experience.

Modest bandwidth savings have given disproportionate latency savings. Latency gains have varied from nearly 7 times to almost 30 times the calculated gain. Such variation should not be too surprising: the calculated gain did not take into account the effect of traffic reduction on other network traffic. In particular, removal of hot spots has a disproportionate effect on reducing latency.

The overall effect, as seen by a user, could include lower annoyance with unpredictable behaviour, e.g., reduction of jitter and other artifacts of congestion. If the latency savings were to translate into a real-world scenario, VMC would be worth implementing.

5 Conclusions

VMC is a promising idea, and a potentially implementable instance of the broader information mass transit (IMT) idea. The version we have investigated here could be realised with simple changes to web applications. Clients (browsers) need to be aware that they should respond to a portion of a data-object (given in the HTTP response codes) by requesting the rest of the object. Alternatively, proxy caches could be used to hide this extra step from the clients, but the costs and benefits of the alternatives are still to be investigated.

The remainder of this section summarizes our results, and proposes further work. Finally, we conclude by considering the potential of both VMC and IMT.

5.1 Summary of Results

Our results show that even with relatively modest bandwidth reduction, VMC can achieve significant latency gains. While the most significant gains are under high load with a high degree of overlap, a large improvement in latency was also seen where there was a high degree of overlap with a light load, or a low degree of overlap with a high load. Particularly in the cases of high overlap, the gains smooth out the cumulative latency graph; this effect is clearest in the case of high overlap and low load. The likely effect as seen by users would be a reduction of artifacts of congestion, such as short-term spikes in latency.

The pathological cases illustrate that the effect on a network with no overlap is insignificant (overhead of less than 1%), while a very high overlap on a highly loaded network, as would be expected, shows VMC to best advantage.

5.2 Future Work

Further work on IMT includes investigation of implementation issues for SAVoD, and investigation of further application of the principles in other areas.

We further propose to investigate areas where VMC can be implemented transparently, and modifications to standard protocols where it cannot be implemented transparently. Specifically, it would be useful to investigate simple alterations to proxy caches to hide VMC from clients, as well as extensions to HTTP which would define behaviour for VMC-aware clients.

More detailed modeling of network traffic would also be useful, to make clearer what the sources of the latency gains are. Insights from such measurement could lead to improvements in VMC, or in other approaches to latency reduction or congestion control.

5.3 Potential of IMT and VMC

VMC has promise. Our initial implementation made it possible to measure the trade-off between benefits and extra costs of VMC. In all cases measured, except the contrived case with no overlap, benefits were significantly better than the cost. With no overlaps, VMC added under 1% to latency, significantly less than the worst gain of 6.88%. More significantly, we found that small reductions in bandwidth could result in significantly bigger gains in latency – much greater than would be predicted by simply calculating the change in transmission time for the reduction in traffic. This finding emphasizes the potential for VMC to reduce hot spots resulting from congestion.

In general, IMT is worth exploring. As Internet bandwidth scales up, traditional models of communication very quickly result in loss of the benefit of new bandwidth. Applications like video on demand are notoriously difficult to scale up, and most proposals have called for very complex hardware and software. Real-time applications, such as web-based TV or radio, are strong candidates for VMC, since they eliminate the need for patching in missed content.

We believe that a new approach is called for, and IMT (including its particular manifestations, SAVoD and VMC) attempts to address this need.

References

1. Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Stephen Williams, and Edward A. Fox. *Caching Proxies: Limitations and Potentials* [online]. December 1995. Available from <http://ei.cs.vt.edu/~succeed/WWW4/WWW4.html>.
2. B Andrew and P. Machanick. The virtual multicasting approach to bandwidth conservation. In *Proc. SATNAC 2000*, Somerset West, South Africa, September 2000. published on CD.
3. B Andrew and P. Machanick. Virtual multicasting as an example of information mass transit. *South African Computer Journal*, (26):252–255, November 2000.
4. S. Deering, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei. An Architecture for Wide-Area Multicast Routing. In *Proc. ACM SIGCOMM Conf. on Communications, Architecture and Protocols*, pages 126–135, 1994.

5. S. Deering, D.L. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei. The PIM architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking*, 4(2):153–162, April 1996.
6. S. E. Deering. Host Extensions for IP Multicasting. RFC 1054 [online]. May 1988. Available from <ftp://ftp.rfc-editor.org/in-notes/rfc1054.txt>.
7. S. E. Deering and D. R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, February 1990.
8. A. Feldmann, A.C. Gilbert, P. Huang, and W. Willinger. Dynamics of ip traffic: a study of the role of variability and the impact of control. In *Proc. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 301–313, Cambridge, Massachusetts, United States, 1999. ACM Press.
9. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP 1.1. RFC 2616 [online]. June 1999. Available from <ftp://ftp.rfc-editor.org/in-notes/rfc2616.txt>.
10. Jozsef Kadlecik, Harald Welte, James Morris, Marc Boucher, and Rusty Russell. The netfilter/iptables project [online]. 2003. Available from <http://www.netfilter.org/>. last accessed February 2003.
11. Brian Neil Levine. *A Comparison of Known Classes of Reliable Multicast Protocols*. Master’s thesis, University of California, Santa Cruz, 1996.
12. Huadong Ma and Kang G. Shin. Multicast video-on-demand services. *ACM SIGCOMM Computer Communication Review*, 32(1):31–43, January 2002.
13. P. Machanick. Design of a scalable video on demand architecture. In *Proc. SAIC-SIT ’98*, pages 211–217, Gordon’s Bay, South Africa, November 1998.
14. P Machanick. Streaming vs. latency in information mass-transit. *Computer Architecture News*, 26(5):4–6, December 1998.
15. Sridhar Pingali, Don Towsley, and James F. Kurose. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. In *Proc. 1994 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 221–230, Nashville, Tennessee, United States, 1994. ACM Press.
16. J. Postel and J. Reynolds. File transfer protocol (FTP). RFC 959 [online]. October 1985. Available from <ftp://ftp.rfc-editor.org/in-notes/rfc959.txt>.
17. Harrick M. Vin Renu Tewari, Michael Dahlin and Jonathon S. Kay. *Beyond Hierarchies: Design Considerations for Distributed Caching on the Internet*. Technical Report TR98-04, The University of Texas at Austin, 1998.
18. A. Rousskov and V. Solokiev. On Performance of Caching Proxies [online]. August 1998. Available from <http://www.cs.ndsu.nodak.edu/~rousskov/research/cache/squid/profiling/papers>.
19. Andrew S. Tanenbaum. *Computer Networks*. Prentice-Hall, fourth edition, 2003.
20. B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *Proc. 21st Annual Joint Conf. of IEEE Computer and Communications Societies*, volume 3, pages 1366–1375, New York, June 2002.

Acknowledgments

We would like to thank The Internet Solution for providing logs on which our preliminary work was based. Logs for the results reported here were provided by Computer and Network Services, University of the Witwatersrand. This work has been supported by the National Research Foundation in South Africa.