

Stepping on the WAN Accelerator

Wes Simpson
Telecom Product Consulting

July 2008



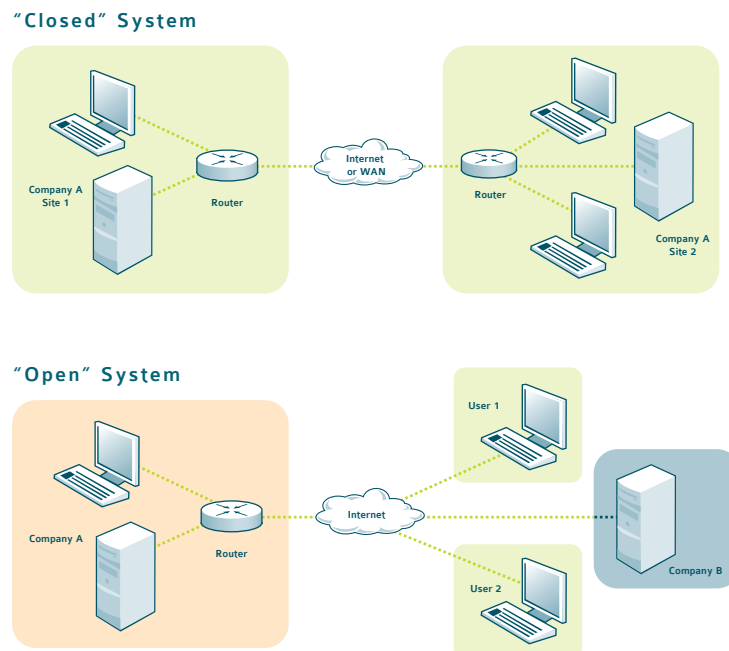
The whole idea of web and WAN acceleration has a lot of appeal, on both an emotional and on an economic level. After all, who wouldn't want to do things quicker on the Internet and save money at the same time? However, as I quickly discovered when I began to investigate this market space, there is a bewildering variety of technologies available, and a large number of suppliers all claiming that their solution is the best. So the challenge is how to make sense out of all these claims.

There are many applications where accelerating file transfers over wide area networks is of great benefit to companies and users. In content delivery, faster file downloads can mean quicker fulfillment for content providers and faster start times for users. For moving massive, professional-quality media files between content owners and producers, accelerated deliveries can mean the difference between over-night sessions and same-day turnaround. Point-to-multipoint file deliveries that are common in government, manufacturing, healthcare, and a variety of other applications will also benefit from acceleration. And there are thousands more applications that could benefit from increased file transfer speeds. (Are you listening, web video providers?)

Open and Closed Systems

To really get a handle on this marketplace, it makes sense to divide it up into several different applications. One big distinction in the acceleration market is the difference between closed and open applications. By this, I mean whether the application involves data transfer between different parts of the same company or an organization (a closed system) or if it involves data transfer between an organization and entities that may have little or nothing to do with that organization on a regular basis (an open system), as shown in Figure 1. Why is this distinction important? Well, when both the source and the destination of a data flow belong to the same company or organization, and if these data transfers take place on a regular basis, then it makes sense to use a double-ended solution, where acceleration hardware and/or software is installed on both ends

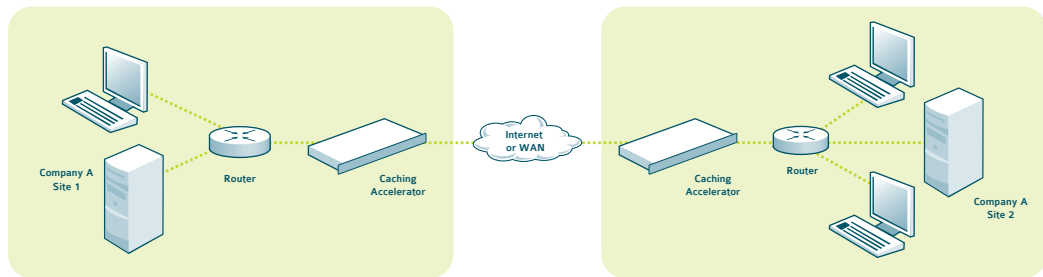
Figure 1



of the link. When this type of solution is used, there are a lot of techniques that can be used to speed up file transport, including local caching at both ends, sending only the changes to files that have been previously transmitted, or using some form of compression to reduce the size of the files in transit as shown in figure 2.

Figure 2

Double-Ended Solution



The challenge is much greater when files are being sent using open applications between unrelated parties, because the sender cannot assume that the receiver has any useful prior knowledge of the file's contents. In this instance, one method that designers have used could be called "catch" software installation, where the receiver is required to download some type of applet that is used to receive a specially formatted file from the source. For example, the catch software could be used to decompress data that was compressed by the sender, reducing the number of bits sent and speeding up the transfer. Of course, this comes with a string attached – the receiver must be persuaded to download (or accept delivery of) the software, which can be a headache for users with tightly controlled configurations or who are using alternative operating systems.

Possibly the most interesting method that I have heard of for WAN acceleration does not require a closed system or catch software. Frankly, I was a bit skeptical when I first heard of it. If you eliminate any kind of compression or file differencing and you also eliminate any downloads to the receiver, how is any kind of acceleration possible? As I learned, the secret has to do with changing the way that file transfer works. The culprit is really with TCP (Transmission Control Protocol) and the way that it is normally implemented. TCP is used every time a user visits a website, sends and receives e-mail, or transfers files between the user's device and a server located down the hall or in another country. So, my education had to start with learning how TCP really works.

Issues with TCP

TCP has been part of the very fabric of the Internet since 1981, and is used billions of times every day for all types of file transfers, both large and small. TCP's basic role is to ensure accurate delivery of files from a source to a destination (such as a server to a client) with every byte intact. This is important for all types of files, such as software downloads, documents, spreadsheets, and audio/visual content files. If bytes are lost during transmission due to errors or network congestion, many of these files become useless.

To achieve perfect transmission, every time a TCP client receives a packet, it sends an acknowledgement back to the sender. As soon as the sender receives the acknowledgment, it knows that it can send another packet. (In reality, several packets can be in transit at any time, provided the receiver has a large enough buffer to receive multiple packets.) Both the sender and the receiver keep track of the sequence numbers that are sent with each packet and each acknowledgement. As more packets are successfully delivered, the sender can slowly increase the rate that it sends packets.

If a packet is lost along the way, the receiver sends an acknowledgment indicating that it did not receive one of the packets that it was expecting. Whenever the sender is notified of a lost packet, it sends a replacement. In addition, most senders will reduce the packet-sending rate when packets are lost, in a process called *flow-control*.

The basic concept behind flow-control is that data sources in a well-behaved network should not try to send data faster than the network can handle it. This is an excellent principle, but unfortunately, because most TCP implementations use an algorithm called AIMD (for additive increase/multiplicative decrease), their behavior is not well suited for large file transfers. AIMD operates by slowly increasing the packet-sending rate while things are going smoothly (i.e. no lost packets) and rapidly decreasing the rate once packet losses occur. Typically, each increase is a small step (adding a small, fixed amount to the bit rate) while each decrease is large (cutting the bit rate in half). For small files, such as a 10-kbyte web page, this isn't a big issue. For large files (such as a 7-Gbyte DVD-master file) this behavior can greatly increase the time required to deliver a file over a network. In addition, AIMD continually pushes data flow rates to the point where they overflow the network, which can not only harm the high bandwidth flows, but can also hurt other flows on the network that are using the same network paths.

The Boiling Pot

To help understand TCP's default flow-control behavior, an analogy might be appropriate. Picture a cook making a pot of stew using a stove that has only two settings – OFF and ON, and no way to measure the temperature of the stew. The cook puts the pot on the stove and lets the temperature of the stew increase slowly until it reaches the point of boiling over. As soon as this happens, the cook removes the pot from the stove, lets the pot cool down, and then places it back on the stove for the cycle to begin again. This process is almost identical to TCP's flow control, where packets are sent at an increasing rate until loss begins to happen, at which point the rate is cut dramatically and then begins to slowly increase again.

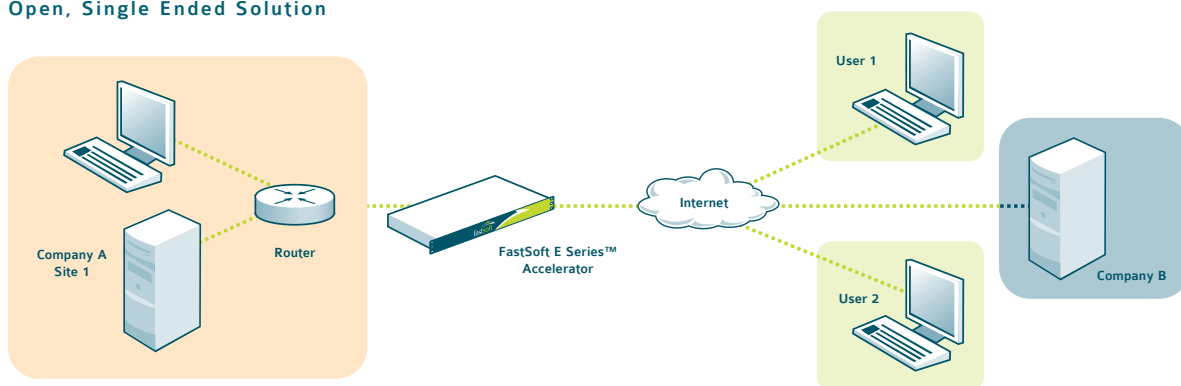
Clearly, this isn't the best way to cook. First of all, the stew is subject to a constant cycle of heating and cooling, which is not terribly efficient. Second, it is easy to see how a few random spatters could be mistaken for boiling over, which could trigger unnecessary cool-down cycles. (This is exactly what happens when a random packet loss triggers a multiplicative decrease in the packet sending rate.) Finally, this process makes a lot of demands on the cook and ends up taking longer to finish the stew.

Fixing TCP

Back to our original issue – how do you speed up TCP in an open system without forcing the receiver to modify their behavior or system in any way? As it turns out, a company named FastSoft has developed a technology called FastTCP to do precisely that. FastSoft has released a product called the E-Series Accelerator that is now on the market that incorporates this technology, as shown in Figure 3. I was very intrigued to see how it works. The fundamental innovation behind FastTCP was the recognition that AIMD was the culprit, not TCP's basic behavior that ensures complete and accurate file transfers. FastTCP works by measuring the amount of time it takes for data packets to be sent and acknowledged, as well as tracking any lost packets. This gives FastTCP's flow-control mechanism much more information to work with, thereby permitting intelligent flow control. With this extra knowledge, FastTCP can make sure that each data flow optimally uses the available bandwidth on the network, while avoiding the overflows and constant up/down cycles caused by AIMD.

Figure 3

Open, Single Ended Solution



FastTCP supports Internet and WAN acceleration with two big advantages for users. First, no change whatsoever is required on the receiver side. Standard TCP handshaking, which has been implemented in dozens of operating systems and millions of devices, will work without a single change. Second, no changes are required to the server software or hardware, because the accelerator device is connected between the server and the Internet (or any other wide area network for that matter). This means that the technology can be installed without having to go through the painful process of upgrading or configuring server software.

Let's take a look at how this all works. As soon as the file transfer begins, the accelerator goes to work by monitoring the outbound packet stream and measuring the amount of time required for acknowledgement packets to be received. The accelerator also begins to buffer packets from the server by sending its own acknowledgements in place of the client's, thereby speeding up the transfer of the file out of the server. Once acknowledgements start to arrive from the client, the accelerator executes an algorithm to determine the speediest rate that can be used to deliver packets to the client without overflowing the network, and begins sending packets from its buffer to the client at this optimal speed. If packet losses do occur, the unit can quickly determine if the losses are simply random or if they are being caused by congestion, and then takes the appropriate action. Note that in this entire example, all of the responsibility for acceleration falls on the E-Series, not on the server or the client, which both experience speedier, more efficient file deliveries.

In a sense, FastTCP works like a stove that can simmer a stew at exactly the right temperature – just enough heat to keep it cooking nicely without any danger of boiling over. FastSoft provided some data about the amount of acceleration that can be achieved with their product. In one test, a 100 Mbit circuit that runs from Connecticut (no, not in my house, unfortunately) to Los Angeles was used to transport a 100 Megabyte file. Without the accelerator, the total transit time was 42 minutes, due to the inefficiencies of AIMD and a non-zero packet loss rate. (Packet losses will be present on essentially all long-haul telecom connections.) With the E-Series Accelerator inserted at the source, the transit time dropped to 3 minutes, representing a factor of 14 improvement.

Accelerating to the Finish

Of course, there are downsides to FastSoft's approach. First of all, users need to purchase the accelerator device for each location where they host content. The good news is that one box can handle the output of multiple servers simultaneously, so that might not be too big an issue. Second, this technology only provides a benefit in one direction only – if a receiver is trying to upload a file back to the sender, no acceleration will occur (but the file will still go through).

Overall, FastTCP seems like a pretty valuable contribution to the field of Internet and WAN acceleration – particularly because it remains completely hidden from users, avoiding the inconvenience and commitment that most other solutions on the market require. In my opinion, anything that can be done to speed up file transfers without forcing me to purchase any new hardware or download and install yet another applet on my PC is a great thing, and FastSoft seems to have the right approach.

About the Author

Wes Simpson is President of Telecom Product Consulting, which he founded in 2000 to provide high quality research, marketing, business development, and product management services to companies wishing to capitalize on the expanding market for high performance video telecommunication products and services. Wes is a frequent television industry speaker at events such as NAB, IBC, SMPTE, and VidTrans, and he is a regular columnist for TV Technology. He has written two books published by Focal Press: **"IPTV and Internet Video"** in 2007 and the second edition of **"Video Over IP"** which was published in 2008. Please feel free to contact Wes at wes.simpson@gmail.com.

About FastSoft

FastSoft (fastsoft.com) was launched in 2006 to bring the benefits of Internet acceleration technology to the media and entertainment industry and other enterprises that depend on the transfer of large files. FastSoft accelerates video up to 15x for long-distance production collaboration, digital delivery to customers, and superior broadband viewing in the home. Large files - including CAD, imaging, business information, and software – can see an improvement of up to 30x faster. Unlike other acceleration solutions, FastSoft's products are single-box solutions that require no hardware or software on the receiving end. The company's core technology, FastTCP™, is based on research originally developed by FastSoft's founders at the California Institute of Technology's Networking Laboratory (Netlab).