Remote Access Response Time Blues

Robin Layland

Application accelerators come to the rescue.

One of the big complaints from remote sites is always response time. Users constantly ask network managers to improve response time, to make it more like headquarters. The traditional remedy, "throw bandwidth at it," works in many cases. Doubling the speed of the line does reduce the amount of time it takes to transmit the packets and is generally the best solution when the link is running at a high utilization.

But what is a network manager to do when link utilization is not the problem, and the remote site users are still unhappy with their response times? If the users are experiencing slow response time and the link utilization is low—under 40 percent—then adding more bandwidth will only marginally improve their response time.

If there are only a few users at the remote site, throwing bandwidth at the problem is also unlikely to improve the situation because the real problem is the TCP connection limiting the amount of data that can be sent. Adding bandwidth doesn't address this problem, it just makes the link utilization percentage go down. Plus, throwing bandwidth at the problem every time someone complains is expensive.

Technology, in the form of application accelerators, can help improve response time even when there is no crisis. Application acceleration helps by addressing non-bandwidth-congestion problems and by taking action to speed up the entire process.

What situations are candidates for application accelerators? Does any of the following sound like your network? If so, consider an application accelerator.

- Remote users complain about poor response time; the response times are higher than expected given the link speeds; the links are running at a low utilization and you determine that the application is not the problem.
- Files take forever to retrieve and a datacenter consolidation effort was just finished, where local file servers were moved to the datacenter.
- Backups and restores take too long. While slowness may be expected, given the amount of data being transferred, the business units find it unacceptable.
- The application group moved, or rolled out an application using HTTP and browsers as the user interface instead of the older client/server interface, and no one is happy about the response time.
- A business unit demands faster response time at a remote location. It doesn't matter that the application is a bandwidth hog such as CAD/CAM, they consider it the network manager's job to make it run faster. Throwing bandwidth would be impractical because of the amount of bandwidth the application requires.
- The remote users are on the other side of the world, and adding bandwidth is considered the last resort.

The first wave of application accelerators to hit the market was called *WAN optimizers*. They compressed the data and manipulated TCP flow control parameters to improve throughput and response time. When they first appeared, the main marketing thrust was getting more out of the link and delaying upgrades. These early features are still important, but the emphasis is now on improving response time.

Application accelerators are the next wave of optimizers. They combine solutions from the first wave with innovative methods. WAN optimization vendors such as Packeteer, Expand, Peribit (recently acquired by Juniper Networks) and Riverbed bring their TCP expertise and improved compression techniques, along with new techniques aimed at file servers.

Another class of products, *application front ends* (see *BCR*, March 2005, pp. 54–59) come from vendors such as Fine Ground, Array Networks, F5, NetScaler and Redline (also acquired by Juniper), who leverage their knowledge of Web applications and HTTP. In addition, players such as Tacit Networks and Cisco provide accelerators aimed at the file server market.

Reducing The Size

Compression has always been the first way to help remote users. Traditionally, compression was

Robin Layland is president of Layland Consulting, a firm that specializes in network architecture and new technology. He has more than 25 years' experience in enterprise networking, including technical and management positions at American Express and Travelers.

Pattern recognition requires a device at both ends of the connection

done by applying an algorithm to the packets, producing a shorter version. This is still an effective way to reduce the amount of data and the size of what is sent. Additionally, sending fewer packets makes it less likely that TCP's flow control will slow the process down.

Traditional compression can reduce the size of a transmission by a factor of 2 to 4, depending on how good the algorithm is. For example, browsers support a standard solution—GZIP. Expect a reduction of 2 to 3 times with GZIP. The advantage to GZIP is that you only need compression equipment at the datacenter—i.e., it's an asymmetrical solution—since the capability comes built into newer browsers.

Getting a better rate requires better algorithms—algorithms not built into the browser. This requires that equipment be placed at the remote site, for a symmetrical solution. Symmetrical solutions can get better reduction: depending on the data, 3 or 4 times, even up to 10 times.

A new type of compression is sparking debate among vendors. The approach takes a string of bits and reduces it to a shorter unique string. The receiver then applies the relevant algorithm and reverses the process. Several vendors, including Riverbed, Expand and Peribit, have come up with a different method that combines concepts from compression with this technique, called pattern recognition.

Pattern recognition allows a large block of data to be reduced to a reference number. Figure 1 shows a simple example. The message "Example of using patterns to make a message shorter" is sent. The acceleration equipment takes the message and breaks it down into parts that are then stored in a table, each item having a reference number. The accelerator then sends the message to the end-station, along with information on how it was broken down.

The first time there is no gain, unlike traditional compression that can reduce the size of any message. The real advantage comes on later messages. When the second message—in this example a little different from the first—is sent, the acceleration can use its table to reduce the message. In this example, only "makes the" part of the original messages needs to be sent. The remainder of the message can use the data stored in the table. The accelerator needs to send only the very short table references.

As the accelerator builds up its table, pattern recognition can greatly reduce the amount of data sent. For example, if two people at a remote site both receive the same email attachment, the amount of data sent for the second person's attachment can be very small, possibly 10 to 100 times less.

The vendors differ on the algorithms used to determine the sizes of the pattern. They also can vary when in a pattern to start. For example, if they store a long pattern, they can tell the other side to start in the middle of the pattern. This increases the chance the accelerator can use a pattern it already has, saving the bandwidth it would take to send a new pattern down. This technique can also be applied to messages compressed by the older approach, further reducing the size.

FIGURE 1 Pattern Recognition Example

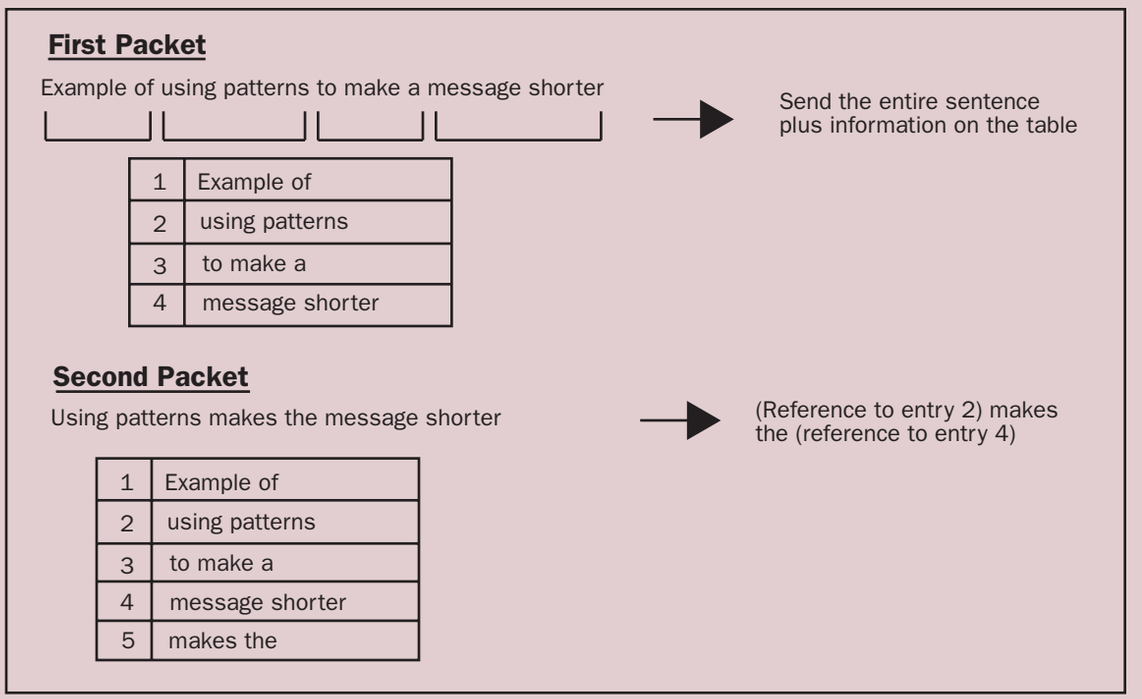
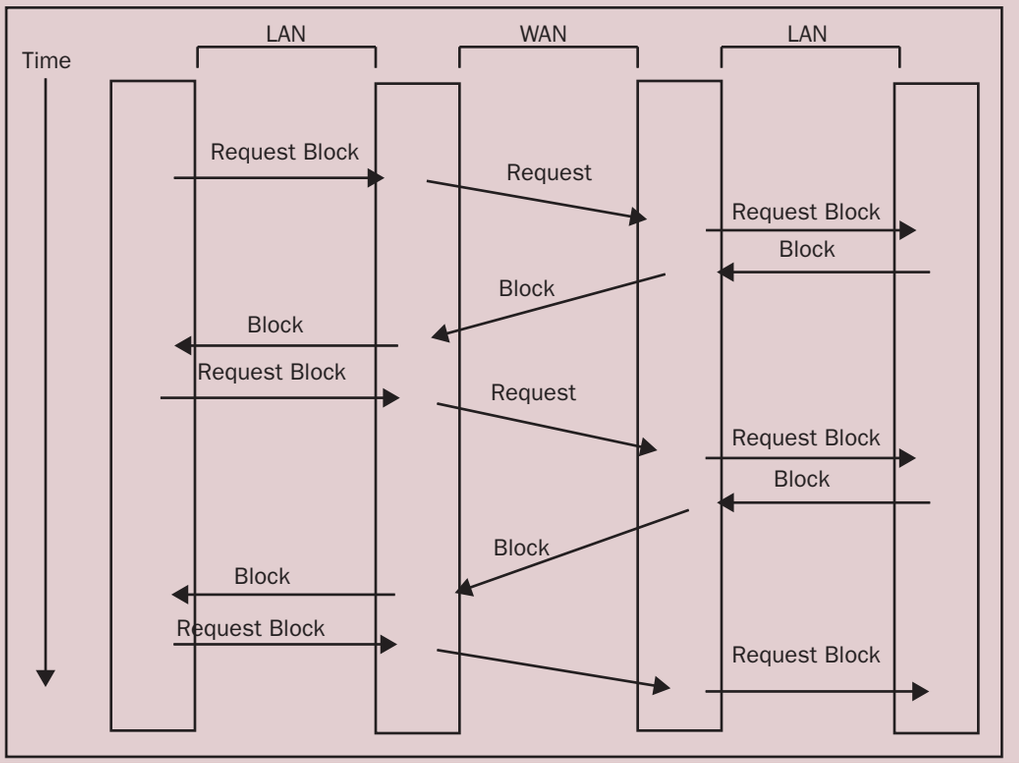


FIGURE 2 Normal Operation For File Protocols



Pattern recognition requires a symmetric approach; an accelerator is needed at both sides. Storing patterns also requires memory. Traditional compression only required a fast processor to keep up with the data flow, but pattern recognition requires a fast processor plus a lot of storage. Vendors such as Expand use a large amount of memory, up to 4 gigabytes, while other vendors such as Riverbed use memory and a disk. At this point, there is not enough independent research to know which approach, disk or memory, is the best, and it more than likely depends on the type of data being sent.

Files

Datacenter consolidation can negatively affect remote users. Traditionally, file servers were located near the user and were accessed over a high-speed LAN, so there was very low latency between client and server. A low latency environment means the design did not worry about protocol efficiency. The number of blocks retrieved at one time and the size of the block of data being retrieved with a single Get were unimportant. Thus the protocol sends a Get Block, waits for the block and then issues another Get only after receiving the first block. This is the equivalent of a fixed window size of 1, which is a very inefficient flow control process if a WAN is involved.

Datacenter consolidation changes the equation. The file servers are moved to the datacenter, which makes sense because the cost of maintaining the server at the local site is much greater than at the datacenter. The problem is that the time it takes to send the file is now unacceptably long, thanks to the WAN-inefficient application protocols, with their window size of 1.

The application acceleration vendors are addressing this problem. Compression helps, especially the pattern recognition type, but it can't solve a fundamental problem: Propagation delay, the time it takes the electrical or optical signal to travel from the datacenter, often is the largest part of the equation, and there is no way to shorten propagation delay between two sites.

Normally, this problem would not be so bad, because TCP allows multiple blocks to be in transit; but the aforementioned protocols from Microsoft and other file server vendors are not that intelligent.

The problem is shown in Figure 2. In the time shown in the figure, only two blocks of data are received by the end-user.

The application acceleration vendors have addressed the problem by intervening in the process at both ends. First the accelerator intercepts the Get command and sends it on to the accelerator at the datacenter (Figure 3). The datacenter accelerator then sends GET on to the server, but doesn't stop there; it turns the Get Block into the equivalent of a Get File command. It understands from past Gets for the file that there are many more blocks involved, so as soon as it gets the first block back from the server, it sends that block on to the accelerator at the far end *and* immediately asks for the next block at LAN

HTTP creates new problems for remote sites

speed. It doesn't wait for the user to ask for another block, but starts asking on their behalf, to speed the process up.

The accelerator at the remote site is constantly receiving the blocks and has them ready for the users, making the process more LAN-like. In the example shown, the accelerator has transmitted five blocks of data to the end user, while before only two were received. In many situations, the results can be even better.

The accelerator also increases efficiency by imposing its own protocol between the two accelerators. The vendors are also developing additional improvements by caching files at the accelerators. This eliminates the need to even send the file across the WAN—it can be retrieved from the remote-site accelerator.

In cases where the file has been changed, the accelerator gets the file again and compares it to the version it already has on disk and then sends only the changes. Additional improvements are gained by applying the pattern recognition technique outlined above.

The vendor community has split on two slightly different approaches. One approach, called Wide Area File Server (WAFS) proxies the file server at the end-user's location. This is the approach taken by Cisco and Tacit Networks. The accelerator acts like the file server back at the datacenter. This means the accelerator has to be involved in locking and unlocking the files and making sure users don't make simulation changes to the files stored on the local accelerator. The benefit of this approach is that it gives the accelerator more flexibility to push and keep the file near the end-user.

A second approach is taken by vendors like Peribit and Riverbed. The file server is in the datacenter, and the devices only accelerate getting the file to the local users. They use their pattern recognition techniques to compare changes in the file so that they don't have to resend the entire file every time.

HTTP And Web Applications

Applications are increasingly using HTTP, which has the advantage of being a standard and universal interface, but can also create new problems for remote sites. One of the advantages to HTTP for accelerating response time is that browsers have caches, so objects that reappear on a page don't always need to be retransmitted, which speeds up delivery.

On the negative side, many applications do not take advantage of this feature. An additional problem is that each object on the page uses its own TCP connection. This means that the connections tend to be short lived, never allowing TCP's flow control to reach its full potential. On top of these problems, HTTP can use a large number of back-and-forths to complete an action, slowing down response time.

FIGURE 3 Accelerator Speeding Up Retrieving A File

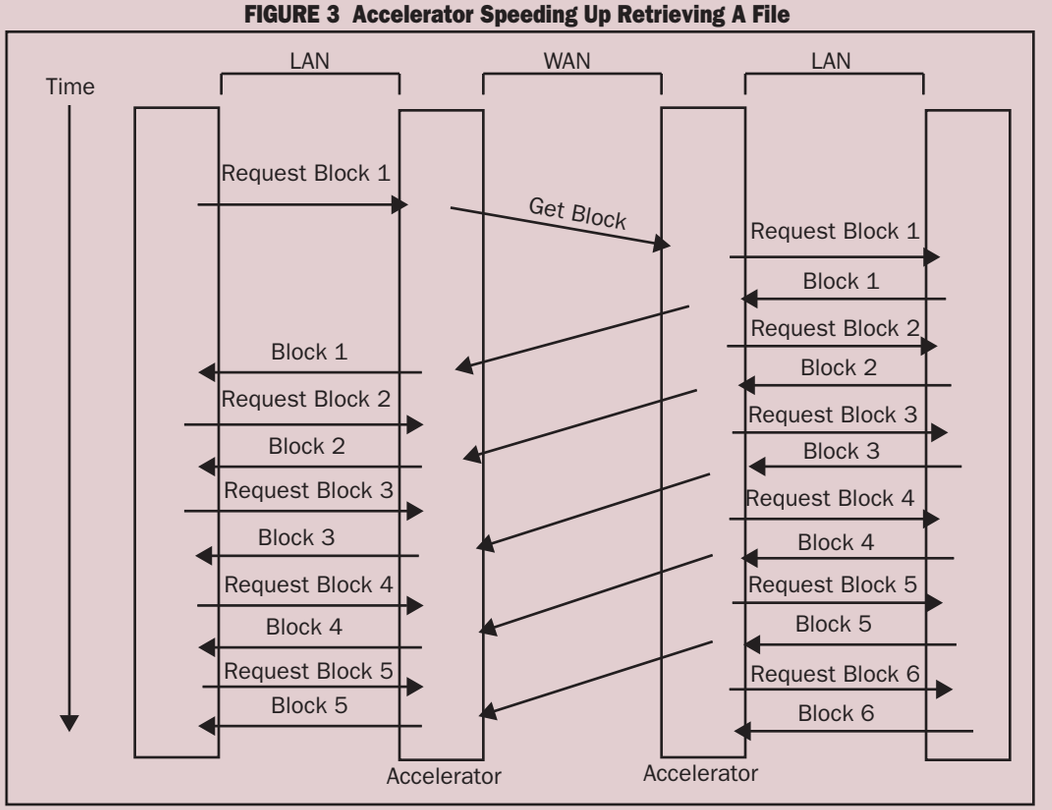
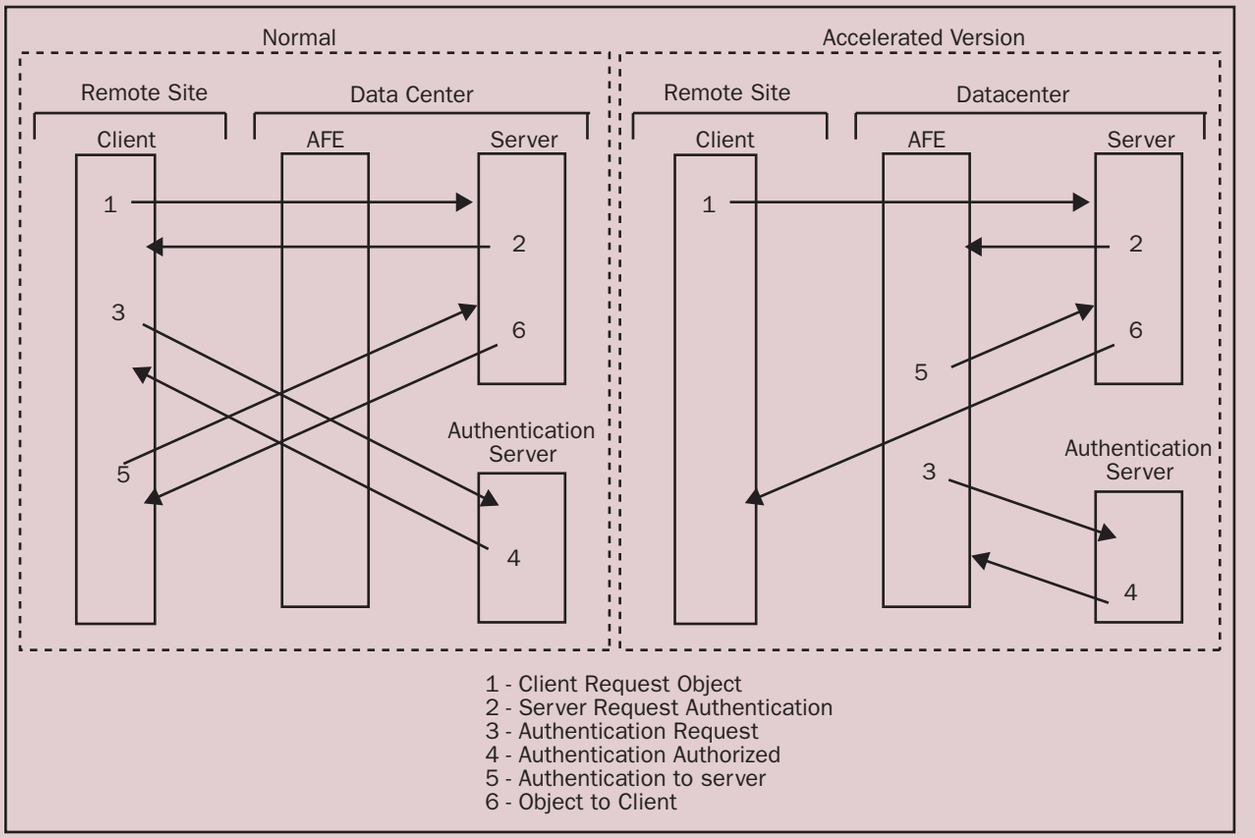


FIGURE 4 Example Of An HTTP Transaction That Needs Authentication



Application accelerators from vendors such as Array Networks, NetScaler, Fine Ground, Redline Networks, Radware and F5 Networks have addressed these problems. The accelerators keep track of the object already cached at the user's browser. When the server sends an object that is already at the browser, the datacenter-based accelerator tells the browser to use the object in its cache, saving the WAN bandwidth.

There are some variations on this that make it even more efficient and faster. If the browser requests an object the accelerator has cached, the accelerator sends the object from its cache, saving on the time it takes to get it from the server. If there is an accelerator at the remote site, the accelerator there can send the object from its cache, saving even more time. Almost all accelerators that have this feature apply it to static objects. Some of the more advanced accelerators can apply the technique to dynamic objects.

When the object is not in any cache, the accelerators can still speed up the process. For example, Packeteer learns what objects are on a page the first time it sees the page. If, later on, there is a request for the page, the accelerator knows what objects are on the page and immediately sends out a request for all the objects, without waiting for separate requests for each object. Often, the accelerator will already have asked for and received a given object before the browser would get around to doing so—speeding up the entire process. This is especially useful for dynamic objects that can't be cached.

Another problem for HTTP applications is TCP's slow-start algorithm, which is meant to prevent a connection from sending too much data at the beginning, giving it a chance to find out if the link is near congestion. If there is no congestion, then the connection slowly opens up. The original assumption was that the TCP connection would last a long time, and this was generally the case before HTTP. If HTTP were operating under the old slow-start assumption, it would have used just one connection for all the data on the page. Instead, HTTP upset this assumption by using a large number of connections to deliver the information on a page, generally never giving slow-start time to reach an efficient data rate.

Accelerators overcome the problem of short-lived TCP connections by either modifying the TCP flow control, speeding it up or changing the way the connection works between two accelerators. Accelerators from vendors such as Expand, Packeteer, Peribit and Riverbed modify TCP's flow control to accelerate the slow-start processing, allowing more data to be sent faster.

When there is an accelerator at both end points, they take it further by multiplexing the individual

connections into a permanent TCP connection between the two accelerators. This overcomes the slow-start process, because the connection between the two accelerators is already operating at peak efficiency.

Another way accelerators speed up HTTP is by shortening the trips. This is done by understanding the HTTP protocol at the application level and then actively intervening in the session to make it more efficient.

An example best shows how accelerators are getting creative in shortening HTTP transactions. The left part of Figure 4 shows a browser requesting an object that has to be authenticated to make sure the user is allowed to receive the object.

The flows needed to complete the process are shown in the figure with the number indicating their sequence. The browser first requests the object (1). The server then tells the browser it needs authentication (2). The browser then sends an authentication request to the authentication server (3). The authentication server then approves the request and sends the required authentication information back to the browser (4). The browser then forwards this to the server (5) and the server then starts sending the object to the browser (6). It takes three round trips over the WAN from the user at the remote site to servers in the datacenter to start getting the object (#6 just shows the first packet of the actual object being sent to the user's browser).

Fine Ground's accelerator shortens this example to only one round trip over the WAN, and keeps the remainder within the high-speed LAN at the datacenter. The right side of Figure 4 shows how it's done.

The accelerator intercepts the server's authentication request going to the client (2). It instead knows the request (2) has to eventually go to the authentication server and sends it directly (3) to the authentication server, acting as the user's browser. When the authentication server sends back the OK (4), the accelerator again short-circuits the process and sends the OK directly to the server (5), again saving a WAN trip. The server then starts sending the object to the user's browser (6). Accelerators are increasingly adding expertise such as this example to speed up HTTP.

Conclusion

Application acceleration is a fast-moving area in which we can expect both technical improvements and more attention from the market. Juniper Networks' acquisition of two of the leading companies in this area, Peribit and Redline, shows that the offering in the market will change over the next year.

The only constant is that accelerators will improve and become more important to enterprises in the future and will comprise a standard part of remote site equipment□

Companies Mentioned In This Article

Array Networks (www.arraynetworks.net)
Cisco (www.cisco.com)
Expand (www.expand.com)
F5 (www.f5.com)
Fine Ground (www.fineground.com)
Juniper (www.juniper.net)
NetScaler (www.netscaler.com)
Packeteer (www.packeteer.com)
Peribit (www.peribit.com)
Radware (www.radware.com)
Redline (www.redlinenetworks.com)
Riverbed (www.riverbed.com)
Tacit Networks (www.tacitnetworks.com)

Accelerators can cut transactions from multiple WAN trips to a single traversal