

Citrix and Terminal Server Network Performance Tuning and Troubleshooting

Brian Madden

February 2005

This paper focuses on how certain characteristics of your network can affect your server-based computing environment and what you can do about it.

Unfortunately, every network is “bad” at some point, and most of the people don’t think they have the ability to do much about that. This paper will show that you can change the behavior of your Citrix MetaFrame and Microsoft Terminal Servers to “tune” them for many different network environments. It is possible to deal with low bandwidth, high latency, jitter, packet loss, data corruption, dropped packets, and low MTU limits.

I have written extensively in the past about how you can tune the performance of your Citrix MetaFrame or Microsoft Terminal Servers. However, I think I've only been focusing on part of the performance challenge.

The part that I've been neglecting is the network. As far as overall performance of your environment is concerned, it doesn't matter how well you tune the processor, memory, and application performance of your servers if you have a bad network. A poorly performing network can kill Terminal Servers that work fine locally.

Unfortunately, every network is "bad" at some point, and most of the people reading this paper probably don't have the ability to do much about that. Therefore, we'll focus on how certain characteristics of the network can affect your server-based computing environment and what you can do about it.

How can network characteristics affect server-based computing?

I think it's safe to say that when thinking about network performance, most people focus on two network characteristics: bandwidth and latency.

In order to really dig into it though, we also need to look at a few more properties of a network: packet loss, out-of-order sequencing, jitter, MTU size, and link errors.

Let's step through each of these six characteristics and look at how they can affect a Citrix MetaFrame or Microsoft Terminal Server environment. Then we'll look at how you can test for these and what you can do about them.

Bandwidth

Bandwidth is the most commonly-referred to networking characteristic. It simply defines the amount of data that can be transferred from one computer to another in a certain amount of time. This leads to the classic question:

So how much bandwidth does Citrix take?

The classic answer: *That depends. How much do you have?*

While this is somewhat of a joke, it's actually rooted in truth. Most applications will take as much bandwidth as they possibly can. In general, more bandwidth is always better, although there are limits to what's practical.

Another variation of the classic question: *I have xxx bandwidth. How many Citrix users will that support?*

How does this affect Citrix?

In order to answer these questions and understand how the amount of available bandwidth will affect Citrix MetaFrame and Terminal Server environments, it's important to understand how the ICA and RDP protocols work.

Fundamentally, Citrix's ICA and Microsoft's RDP protocols are simple data streams that are sent between the client and the server. If the amount of data moving through the stream is less than the amount of bandwidth that's available, then

everything will be fine and you'll have no problem. However, if your ICA or RDP protocol tries to use more bandwidth than is available, interesting things will start to happen.

But how much bandwidth does ICA or RDP use? There are statements floating around on the Internet that say things like "Citrix ICA uses 15kbps" (or 20k or 30k or whatever). There is one thing that's critical to understand about these statements: *Any statement about ICA or RDP bandwidth usage simply refers to an **average** over some period of time.* The longer the time period, the more accurate the average.

In real life, ICA and RDP are extremely "bursty." If you don't touch the mouse and nothing's happening on the screen, no data will cross the network (except for the occasional keep alive packet or similar). However, if you launch a new application that completely updates the screen, you might see 70k of data move from the server to the client in 1/3 of a second.

If you open performance monitor and view the amount of bandwidth a session takes (discussed later), you'll see many sharp spikes. These "15kbps average" numbers simply come from someone running a session for maybe an hour and then looking at the results. "Hmmm... 6.7MB was transferred over the last hour, so that means this session required 15kbps per second." While it's true that 15kbps was the "average," the session probably actually used maybe 100kbps for 30 seconds (distributed over the hour) and 2kbps for the remaining 59 minutes and 30 seconds.

As a side note, there is a rumor floating around on the Internet (based on an article in Windows & .NET Magazine) that the RDP protocol is a "stream" protocol (like RealAudio, etc.) and that it consumes bandwidth regardless of whether the user is actually doing something. That rumor is absolutely 100% false.

So how much bandwidth does ICA or RDP actually require? It all depends. Of course there are things you can do (discussed later) to "minimize" the bandwidth consumption of RDP or ICA, but that minimization really does nothing more than limit the spikes in data transfer.

Where you'll start to see issues is when ICA or RDP wants to "spike" more than the available bandwidth (either because other data is on the network or because the network limit is hit). When this happens, the ICA or RDP data will be queued on the sending machine until it can be sent. This has the effect of increasing latency (as discussed in the next section). Of course if too much data is queued up then it's possible that the sending computer's network output buffers will become full and then data packets will be discarded (as discussed in the packet loss section later in this article).

Latency

Latency is the amount of time, measured in milliseconds (1000ms = 1 second), that it takes for data to cross a network between two computers. On computers that are connected to the same switch, latency might only be 2 or 3ms. Many connections over the Internet will have latency in the 200 to 300ms range. Connections bounced over satellite communications might have 700 to 1000ms latency.

How does this affect Citrix?

In all cases lower latency is better. In Citrix and Terminal Server environments, having low latency is probably more important than having adequate bandwidth. This is due to the fact that all interactive user activities happen remotely when using ICA or RDP.

Imagine a server-based computing connection with 200ms one-way latency. As a user, you would type the letter “A” on your client device. It would take 200ms for the packet containing the keycode for the letter “A” to arrive at the server. It would take a few milliseconds for the server to process the keystroke and put the “A” on the virtual frame buffer, and then it would take another 200ms for the updated screenshot to get back to your client device. Therefore, the total roundtrip delay would be around 400ms, or almost half a second. This would have the effect of having a half-second “delay” in your typing.

Environments with very low latency will be taken for granted by your users, but as soon as that latency starts to creep up you’ll start hearing about it.

Packet Loss

Packet loss is the term that defines what happens when packets from the sending computer don’t show up at the receiving computer. In server-based computing environments this can be from client to server or server to client. Packets usually get lost when there is a problem with the network or when the network is too busy. To understand how this applies in server-based computing environments, we need to take a step back and look at how TCP/IP packet-based communication takes place in general.

Fundamentally, two nodes on a network communicate by sending a stream of data back and forth. However, since TCP/IP networks are “packet switched,” this stream of data is broken up into a bunch of little pieces, and each of these little pieces arrives at the destination on its own where the destination computer reassembles them to form the original data stream. The exact size of these packets varies, but in most standard Ethernet environments each packet is 1.5kB.

There’s another interesting aspect of TCP/IP communications that’s relevant to ICA or RDP communication. Fundamentally, network communication is viewed as unreliable. There is no guarantee that a packet sent by a computer will actually get to its destination. (Many things could cause this: Collisions, equipment failure along the way, congestion, etc.)

As you can imagine, in an ICA or RDP environment, a lost packet would cause a “hole” in the stream of data. What if that “hole” contained keystrokes or characters to be put on the screen? Clearly, losing packets is a bad thing.

If a packet got lost, the sending computer would need to know about it so that it could resend the missing data. In the TCP/IP world, this is done via a special packet type called an “ack” (or “acknowledgement”). It works like this: The sending computer sends a bunch of packets to the receiving computer. Each packet contains a sequential serial number (called a “sequence”). Every so often the receiving computer sends an “ack” packet back to the sending computer with the number of the most recent sequence that’s been received in full. Therefore, if any of the

sending computer's packets got lost along the way, the receiving computer would never send out the ack, so the sending computer knows that it needs to resend the packet. See Microsoft knowledgebase article 169292 for all the gory details of acks, receiving window buffers, and other TCP/IP communication internals.

(Of course an ironic side effect of this is that sometimes the ack packet will get lost in transit, and the sending computer will resend the packet even though the original packets actually arrived.)

So how is this relevant to ICA and RDP? Stay with me...

As we said, if the sending computer never receives the ack packet from the destination computer, the sending computer resends the original packet. However, it only does this a fixed number of times, and if it fails after that the TCP/IP protocol driver gives up and reports back to the application that there was some kind of networking communications error. In Windows, the default number of retries before reporting failure is five, although you can change this in the registry. This is called "TCP Max Retry Count."

Of course computers are very fast, and a sending computer could burn through all five retries in the blink of an eye. To address this, the TCP/IP standards dictate that the sending computer should double the amount of time it waits for an ack response before retransmitting the original packet. For example, if the sending computer waited 2 seconds for the ack before retransmitting, the second attempt of sending the packet would occur after 2 seconds, the third attempt will be 4 seconds after the second, the fourth will be 8 seconds after the third, etc. (Up until the TCP Max Retry Count is hit.)

One of the cool things about TCP/IP is that the exact duration of the retry timer is automatically managed by the operating system and constantly changes for each destination computer. It starts at 3 seconds, but it goes up if the ack packets are taking a long time and goes down if they're coming back quickly. In doing this, each connection is automatically tuned for the actual network performance.

How does this affect Citrix?

Since lost packets must be retransmitted, having a high number of packets lost is effectively like adding a lot of latency. Of course if too many are dropped, then that means that the retransmits are getting dropped too, so eventually your ICA or RDP connection will disconnect.

This issue is usually only seen on unreliable wide-area networks, such as mobile telecom-based wireless networks and remote offices connected via the Internet.

Out of Sequence Packets

Because the data stream is broken up into many pieces which each get to the end location on their own, there's a chance that individual pieces of data might arrive in a different order than they were sent out. This is called "out of sequence" packets because the sequence numbers of the packets will be in the wrong order.

However, the fact that each packet contains a sequence number means that the receiving computer can reassemble the pieces back into the proper order for the receiving application.

How does this affect Citrix?

The good news about packets that are received out of sequence is that this is something that's hidden from the application (the ICA or RDP protocol interfaces), so there's nothing you have to worry about here with regard to server-based computing environments.

Jitter

Of course in the real world, these four aspects of a network usually change over the course of a connection based on current conditions. "Jitter" is simply the term that describes the tendency of a network to have rapidly varying amounts of bandwidth or latency or packet loss. One second it's fine. The next second you suddenly have 600ms latency. Two seconds later you're back to 10ms latency. Then you suddenly drop 10% of the packets. Etc.

How does this affect Citrix?

Some of the aspects of the ICA and TCP tune themselves dynamically, so jitter doesn't affect them. However, other aspects (experience, SpeedScreen, etc.) are set once based on the condition of the connection when it's first made. If the performance of the network then deteriorates, your users will have server-based computing sessions that are "tuned" for the wrong network conditions.

MTU Size

MTU stands for "Maximum Transmittable Unit." In simple terms, it's the size of the biggest packet that can travel across a network. As we mentioned earlier, most Ethernet networks have an MTU size of 1500 bytes (1.5k). However, this can vary by situation. Specifically, most of the mobile telecom-based wireless networks (1xRTT, EVDO, UMTS, GPRS, etc.) have MTU sizes that are much smaller, perhaps in the 400-600 byte range.

How does this affect Citrix?

Let's think about what happens when you have ICA or RDP clients on the other end of a mobile wireless network that has a small MTU size. The "M" in "MTU" stands for "maximum," but there is no minimum size limit. Therefore, if you have a user connecting from a wireless device where the network has an MTU size of 460 bytes, then all the packets that the Citrix or Terminal Server receives will be 460 bytes. So far, so good. No problems yet. (Actually, the initial connection requests from the client don't contain much data, so many of those packets might be even smaller than 460 bytes. A TCP/IP packet is only as big as it needs to be. It is never "padded" to fill the MTU size.)

When the Citrix or Terminal Server starts sending its ICA or RDP data stream to the client device, the data stream will be broken up into packets that are 1500 bytes. (1500 is the default setting for Windows.)

However, at some point that 1500-byte packet will hit the mobile wireless network with the MTU size of 460. That network's edge router will have to break apart that

1500 byte packet into smaller chunks. It will create three 460-byte packets and one 120-byte packet. (I'm simplifying the math here by not counting the TCP/IP stack overhead.)

So how does this affect ICA or RDP performance? In many mobile networks, there is a lot of overhead associated with transmitting packets, regardless of the size of the actual packet itself. In our example we have each packet from the server being broken into four packets for the mobile network. However, one out of every four mobile packets only contains 120 bytes of data, which means all of the overhead needed to transmit a packet is "wasted" on a mere 120 bytes.

Imagine that three 1500-byte packets come from the server. They would be broken into nine 460-byte packets and three 120-byte packets, or twelve packets total. But wouldn't it be better to combine the three 120-byte packets into a single 360-byte packet? That would mean that you would only need to transmit ten packets across the wireless network instead of twelve, a savings of over 8%.

The bottom line is that MTU size changes along the route between your server and ICA or RDP clients can add inefficiencies into your server-based computing environment.

Figuring out which of these issues are affecting you

All networks exhibit some of these six characteristics. That's just a fact of life. The challenge for you is that you have to figure out which (if any) of the six are having a negative impact in your environment and whether it's even possible to make a change that will improve the performance of the session. To understand what we're talking about, let's start with bandwidth limitations.

Bandwidth Limitations

It's pretty easy to figure out how much bandwidth an ICA or RDP session is taking. What's difficult is figuring out whether reducing the bandwidth used will actually translate into better performance for the users. (Of course some could argue that lowering the bandwidth consumption will save on networking costs in the long run, so it's a good thing either way.)

Figuring out whether bandwidth is an issue for you is a multi-step process. The first step is to get some understanding of how much bandwidth a particular session is taking. A quick down-and-dirty way to do this is to use this great \$20 tool called DU Meter (for "Download / Upload Meter). It's this little app that runs in a window in the corner of your desktop. It's made for people who need to track their bandwidth usage for their broadband provider, but I like it because it's cheap, small, simple, and doesn't require a reboot to install or remove.

DU Meter shows live statistics for download and uploads. So, fire it up and launch an ICA or RDP session. You'll see the little display in the corner constantly updating the amount of outgoing and incoming traffic in KB. The best part about it is that it has a little stopwatch function that you can start that will show you the total, maximum, and average data transfer (in KB) in both directions from your client device. I like to drop DU Meter on a desktop and have a user use their session for awhile to get a feel for what I'm dealing with. (Just keep in mind that DU Meter captures *all* network traffic, so if Windows decided to automatically download a

Hotfix in the background while you're running your test then you'll get some weird results.)

You can also use the Performance Monitor MMC snap-in on the Terminal Server to view the total data transferred for a particular session. (Terminal Services Session | Total Bytes | Select the ICA or RDP session you're interested in.)

The challenge here is that once you have this data, what do you do with it?

Latency

The quickest way to check the latency of an environment is to ping the server from the client. However, a more accurate way is to use the Performance Monitor. If you're using Citrix MetaFrame, you'll find a performance object called "ICA Session" with several counters for latency, including the current latency, average latency, and deviation (the difference between the maximum and minimum recorded latency). There is one instance of the ICA Session object for each ICA session, so you need to pick the right session from the list if multiple users are accessing the system and you only want to investigate one.

This counter is accurate, although you should be aware that it only tracks the latency in packets sent from the client to the server. (In MetaFrame Presentation Server 3.0 you can change a registry value that unlocks a new counter called "Active Latency" that can measure latency in both directions.)

Packet Loss

Figuring out whether you are experiencing a significant amount of lost packages is a bit more involved than checking some of the other things, but it's still possible to do and there are a few different ways to do it.

A lot of people will simply execute the ping command with the "-t" option that causes it to run forever and then sit back and look for timed out pings. The problem with this is that real world packet loss rates vary depending on network load, so to get a real test you should ping the server from the client while a user has an ICA or RDP session open. Of course then your ping command would introduce its own load into the environment thereby contaminating the test, so that's really not the best way to do things.

Another option is to use a packet sniffer (my favorite is Ethereal because it's awesome and free) to look for rebroadcast packets, but that can be a fairly involved process.

One of the easiest ways to check for packet loss is to come back to Performance Monitor and track a counter called "Segments Retransmitted / Sec" under the TCP object. Since this is a "per second" rate you're not going to get any aggregates, but any significant jump over zero should tell you that something's not right. (ICA and RDP can easily deal with as high as a 10% packet loss without any real performance problems, so most likely this is not your problem.)

Out of Sequence Packets

It's not really practical to look for out-of-sequence packets per se, but if this happens a lot you'll see it reflected in other areas, such as higher latency.

Jitter

If you're using MetaFrame, it's easiest to look for wild changes in current latency in a session or high latency deviations (both of which are covered in the ICA Session performance object).

MTU Size Limits

As was mentioned previously, most likely the MTU size of all the networks between your clients and server is 1500. You can easily test this with the ping command. Use the "-l" (that's the letter "L," not the number "1") option to specify the length of data to send in the ping packet. (The default is 32 bytes.) For example, you could try to send a packet that is 1500 bytes to see if it makes it. You'll also need to use the "-f" option to mark the ping packet as "not fragmentable"—otherwise it could get broken into pieces and ruin your test.

For example, try the following command: `ping -l 1472 -f yourservername`

You'll see that over the Internet (and in many routed corporate sites) the maximum length you can specify with the ping command is 1472. That's because the IP and ICMP headers add 28 bytes to the packet, bringing the size up to an even 1500. If you get a response that says "Packet needs to be fragmented but DF set" then you know that the length you specified is bigger than the MTU size somewhere along the way. In this case, play a little "guess and check" until you find a length that results in a successful ping. Then, add 28 to that number and you have the MTU size for that route.

Building a Test Lab

Before you can even begin trying to design and implement a solution, you'll need to configure a test lab. This is one of the areas where it really doesn't make sense to try to make these changes on your production network with live users. (Actually, some of these changes are impossible to test on a production network.)

Building a test lab to test the user and design of servers is easy. Just grab some old hardware or a copy of VMware and off you go. But how do you build a test lab that will be relevant for your network troubleshooting and performance tuning? You'll need to build something that can simulate the low bandwidth, latency, and other issues of your production network.

There are several advantages to building such a lab. Specifically, you can find out::

- Citrix performance on different network architectures like Frame Relay, MPLS and QoS.
- How many Citrix users can run from this location with X bandwidth?
- If bandwidth is increased, how will this affect performance?
- Verify and optimize packet shaping and network load balancing solutions before going to production.
- Test the actual quality of experience from the client perspective with live running applications.

Fortunately there are several ways you can implement this test lab this. The easiest and most effective way is with the network simulation products from a company called "Shunra." Shunra has hardware- and software-based solutions that you can

use to simulate your real network's bandwidth, latency, packet loss, jitter, TCP errors, and low MTU sizes.

Shunra's Virtual Enterprise product comes with a Visio-based tool that you can use to model your entire production network. Then you hook up their hardware box to the different segments of your test lab. They even have a product that records up to 30 days of "live" network performance metrics from your production network that will then automatically make your test enterprise act like your production one.

For testing single network segments, Shunra also has a Virtual Network line of software-based products that do basically the same thing on a smaller scale.

Advantages of Building a Test Lab with Shunra Products

- Products simulate your entire environment, including all traffic and all protocols.
- Easy to use with GUI configuration.
- Can use real network data in the lab simulations by capturing production data. (All without using agents!)
- Virtual Enterprise product can simulate multiple segments, so you can test the actual end-to-end multi-site environment.
- Testing environment can be automated and can generate reports to indicate the Citrix performance per scaling network conditions and/or from different emulated remote sites.

If you're using Citrix MetaFrame, there's also a free tool available from Citrix called the "SMC Console" you can use to augment your testing. The SMC Console is actually a sample application that is supposed to show you the kind of cool things you can do with the Citrix Session Monitoring and Control SDK. The SMC Console is part of the MetaFrame Presentation Server SDK 2.3 and newer. This SDK is available as a free download from the Citrix Developer Network (cdn.citrix.com). You have to login to download the SDK, but registration is free.

Once you download the SDK, install it and you'll find the SMC Console as one of the sample applications. (Before version 3.0 of the SDK, the SMC component was an option, so be sure you select that if you want to use the SMC Console.)

Install the SMC console onto a Citrix MetaFrame server and fire it up. The main screen has a dropdown list that lets you to pick which of the current ICA sessions you want to work with. It also shows you all sorts of statistics about that session.

There is a tab called "Display Options" with sliders that you can use to add latency and limit the bandwidth of a particular session.

Advantages of the SMC Console

- It's free.
- Since it's a sample SDK application, it comes with all the source code.

Disadvantages of the SMC Console

- It's a manual configuration only.
- It only works if you're using MetaFrame.
- It only controls ICA traffic and nothing else on the network.

How to Address Each Issue

Now that you have a lab where you can experiment and test out your solutions you can start to think about what changes you'll make to address each of the issues that are affecting you. Let's go through the list one-by-one again, starting with bandwidth.

Bandwidth

If you think that not having enough bandwidth is an issue in your environment, there are a few things you can experiment with:

- Increase bandwidth
- Do something to ICA or RDP to make it take less bandwidth
- Work with the network people to increase the efficiency of the network itself

Increasing Bandwidth

Of course the easiest solution is just to add more bandwidth, right? The problem with this is that it doesn't really address the core issue. Also, if your users are using all of their bandwidth today, what's to say that they won't still use all of the bandwidth even after you increase it?

Of course there's always the chance that you might legitimately have to increase bandwidth. (Maybe you're trying to support 50 remote users over a single dial-up line.) However, adding bandwidth always costs more money, so it's something you should only do after you've tried the other things outlined here.

Limiting the amount of ICA or RDP data on the network

Instead of trying to keep throwing more bandwidth at the problem, why not attack the source? You can do things to the ICA or RDP protocol to minimize the amount of bandwidth it consumes, including:

- Compression
- Bitmap caching
- Queuing keystrokes and mouse data
- Disabling unnecessary virtual channels
- Placing bandwidth limits on certain virtual channels
- Changing session parameters to minimize bandwidth requirements

No of these items is going to make a drastic change in bandwidth consumption, but considering each individually and applying them all as appropriate in your environment can add up to a fairly hefty difference.

Compression

Both the ICA and RDP protocols compress their data to minimize the amount of data that needs to traverse the network. This compression can be as much as 50% for text-based applications (i.e. Word) and 40% less for graphics applications than the uncompressed data streams.

ICA and RDP compression is enabled by default, and there's nothing else you need to do about it. There is sort of an "urban legend" about compression, where people sometimes recommend enabling it. In reality it's always enabled (unless you specifically disable it in an ICA file or RDP file), so people who tell you to enable it don't know what they're talking about.

Some have argued that enabling compression requires extra CPU resources on the client and server to process it, but this is absolutely miniscule in today's world.

Bitmap cache

The ICA and RDP protocols also have the ability to cache portions of screen graphics (called "glyphs") on the client device. Then, if the user navigates to a screen on the server that has portions of it stored in the client's cache, the server can send down an instruction to the client requesting that it pulls portions of the new graphic updates from its local cache rather than the server sending down the same image again.

A client device must have some sort of local storage (hard drive, flash memory, etc.) to make use of the bitmap cache, but other than that there's no real downside to using it. It can be enabled via the client GUI or via an RDP or ICA file.

Queue keyboard data and mouse strokes

This is another option that's enabled by default for both RDP and ICA sessions. With ICA, for example, keyboard typing data is queued on the client device and only sent to the server every 100ms (ten times per second, which is probably faster than people would ever notice). Similarly, mouse movements are only sent to the server every 50ms.

It's important to note that this does not mean that that an update goes from the client to the server every 100 or 50ms. It just means that whenever the user happens to be typing, updates only go out every 100ms.

In environments where bandwidth is a concern, you can increase the client queuing timers so that fewer packets are sent from the client to the server for a given amount of user input. You'll have to experiment with the various settings to see what's acceptable in your environment.

Disable unnecessary virtual channels

As you probably know, the ICA and RDP protocols are made of up several virtual channels. (Printing, port mapping, clipboard integration, audio, etc.) One of the "popular" recommendations floating around on the Internet is that you can save bandwidth by disabling unnecessary virtual channels. While this recommendation is rooted in truth, in reality a lot of people are sad to find that disabling virtual channels doesn't magically save the day.

To know why, you have to know how the ICA or RDP virtual channel infrastructure works. Except for the initial connection (where client and server capabilities and virtual channels are negotiated), disabling virtual channels alone does *not* save any bandwidth.

However, having unneeded virtual channels enabled does increase the chance a user *could* to something via one of them that consumes bandwidth. For example, leaving

the clipboard synchronization virtual channel enabled means that a user cutting and pasting on their local workstation would inadvertently send that clipboard data to the server via ICA or RDP.

Therefore it's probably still a good idea to disable virtual channels you're not using as long as you understand the actual benefit you'll get from this. (Virtual channels are enabled or disabled on a session by session basis, so you can completely customize which users get what.)

Capping Virtual Channels

If you're using MetaFrame, instead of harshly enabling or disabling entire virtual channels, you can take the more eloquent approach of placing specific bandwidth limits on each individual virtual channel. (This cannot be done with RDP.)

Placing bandwidth limits on specific channels doesn't *technically* save any bandwidth. What it does do is limit the ability for one channel to consume too much bandwidth that another channel might need.

Capping the bandwidth of a virtual channel will limit the amount of bandwidth the ICA protocol uses, but at the expense of causing individual channel operations to take more time. For example, if there is 100k that needs to be synchronized to the clipboard via ICA, an unrestricted ICA client might spike to 50kbps for two seconds to perform the sync. However, if you put a 10kbps limit on the clipboard virtual channel, then that same synchronization would consume 10k for ten seconds.

All other things being equal, this virtual channel cap doesn't change the average bandwidth consumption at all. (100k over ten seconds, whether it's 50k for two seconds and 0k for eight seconds or 10k for ten seconds is still an average of 10k per second.) What the cap does do is "flatten" the curve, and in theory let the ICA client perform better when individual spikes occur.

If you want to experiment with ICA virtual channel caps, you can do so with the SMC Console utility we referenced previously.

It's worth pointing out that internally, both the ICA and RDP protocols have priorities assigned to each virtual channel—screen and keyboard data is the highest, printing is the lowest, audio is medium, etc. If you're using ICA then you can tune these priorities on a server-wide basis. RDP's priorities are hard-coded and cannot be changed.

Changing other Session Parameters to Minimize Session Bandwidth

The final thing worth mentioning in this section is that you should keep in mind that you can always change the properties of the session itself to minimize the amount of raw data that will need to flow from the server to the client. You can lower the resolution, decrease the color depth, or lower the audio quality to downsize the data hitting the ICA or RDP client.

Change the Way the Network Deals with Load

All of the settings we've looked at so far affect the ICA or RDP protocol stream itself. However, you can also look at this from the network perspective and adjust

the way the network actually deals with ICA or RDP data. This is primarily done with hardware devices that use the following technologies:

- Packet Shaping
- Compression
- Caching

We'll discuss all of these technologies separately, although most devices on the market today are based on some or all of these technologies.

Packet Shaping

The packet shaping market used to be dominated by Packeteer, Allot, and Sitara, but now there are dozens of vendors in the space. Even though they won't admit it, they all do basically the same thing. They're hardware devices that typically sit on the network near the edge router and intelligently prioritize some traffic while limiting other traffic—all based on business rules. For example, you can configure one of these things so that ICA always has priority over everything else, or so that HTTP and MP3 download traffic is limited to 50% of your outbound connection.

The cool thing about these packet shapers is that as we learned earlier, the faster a sending computer receives the "ack" packets from the receiving computer, the faster it sends the next group of packets. Because of this, these packet shapers are just as effective as slowing down "remote" traffic as they are at slowing down local traffic since they can simply intercept the local outbound "ack" packets and delay them, therefore causing the remote computer to slow down its transmission.

Compression Devices

In addition to packet shaping, some hardware network devices have the ability to compress all of the data that travels through them. (This works with all types of data, including encrypted data, since these devices just do a low-level compression.) These devices work in pairs, typically with one on each end of a WAN link. As the data stream passes through, the first device compresses the data where it's passed on to the edge router. The second device on the remote end receives the data and decompresses it and sends it in its original format to the ultimate destination computer.

These devices are completely transparent to applications, and they work well with ICA and RDP traffic. The amount of compression they can offer varies, but they're definitely worth looking into if you have bandwidth issues and they're usually cheaper than buying more bandwidth.

Caching Devices

Finally, some of these hardware devices also offer intelligent caching capabilities. Again working at a low protocol level, these pairs of devices sit on each side of a WAN link and analyze all traffic moving through. The receiving unit has the ability to cache packet payloads, and both units keep track of all data that passes through.

Then, if the sending end notices data that it has already sent to the receiving end, the sending unit transmits a small instruction tag that indicates the cached location of the original data. The sending unit rebuilds the packets from its cache and sends

the newly-reconstructed original packets on to the destination. This entire process is also transparent to the applications.

Whether network caching devices are effective in your environment really depends on your applications. If most of your remote users are using the same application and that application is built off of several screens, caching can be very effective, sometimes cutting traffic by 40%. On the other hand, if your users are using several random applications then the caching devices might cost you more money than what you can save in bandwidth.

The one caveat to using these types of caching devices is that they cannot be used if you're encrypting the ICA or RDP data with SSL or SecureICA. The reason for this is obvious, in that no two encrypted packets would ever be the same even if they contained the same data.

Latency

As we mentioned previous, latency is probably the biggest headache in server-based computing environments. Having high latency can very quickly kill the usability of a system.

So how much latency is too much? That really depends on the type of application you're using and how smart / patient your users are. Personally I can use a session with 500ms of latency no problem. My grandmother might need something under 200ms though.

There's an interesting theory about latency and usability that was recently posited by Tim Mangan, a seasoned industry veteran and author of some fantastic tools. In a nutshell, Tim suggests that the actually amount of latency you have doesn't really matter. What matters is that that latency is consistent. A session that is continuously at 400ms will be much easier to use than one that's usually at 100ms but that spikes to 300ms every few seconds. (View the full paper at Tim's website, www.turgent.com.)

That being said, there are two things that you can do to minimize the effect that latency has in your environment.

- Free up bandwidth
- Citrix SpeenScreen

Free up Bandwidth

Remember that in many cases, high latency is caused by congestion on the network. Packets get queued up at the sending computer as they wait to be funneled through the limited bandwidth. Therefore, if you have issues with latency in your environment, the first thing you should do is follow the steps in the previous section to limit the amount of bandwidth that each session takes.

It's worth pointing out here that capping specific virtual channels can be particularly effective at smoothing out the spikes that cause a backlog of packets and long latency times.

Citrix SpeedScreen Latency Reduction (SLR)

If you're using Citrix MetaFrame, there is another great tool you can use to combat latency: Citrix's SpeedScreen Latency Reduction (SLR). SLR is technology that's built into MetaFrame Presentation Servers and ICA clients that allows users to experience smooth typing in environments with high latency. Using SLR properly can, for example, allow a user to have a smooth typing experience while writing a Microsoft Word document—even if there is 200, 500, or (gasp!) 1000ms of latency between the server and their ICA client.

Citrix's SpeedScreen Latency Reduction does two things. Firstly, (and most importantly), it provides something called "local text echo." Local text echo allows characters to appear on the ICA client device's screen the instant a user pushes the key on their keyboard.

For example, imagine a situation without SLR where a user is typing a document via an ICA session with 400ms of latency. When the user presses a key, the ICA client sends that key press code to the server. The server receives it, processes it, puts the character into the screen buffer, and sends the new screen image to the client device. However, due to the 400ms latency, the actual character doesn't show up on the user's screen until about a half-second after they first pushed the button. To the user, it would appear that there is a half-second "lag" when typing.

To address this, SLR's Local Text Echo causes the ICA client to behave a bit differently. When enabled, a user pressing a key on their keyboard causes that key code to be sent to the server. However, at the same instant the local ICA client software also paints the appropriate character on the user's screen even though the actual screen painting instructions from the server are bogged down in the 400ms latency between the client and server. Then, once the ICA client finally receives the actual updated screen from the server, it doesn't have to update that character on the local screen since it already put it there back when the user pressed the key. In a sense, SLR's Local Text Echo is kind of like a "pre-caching" of the text. Local Text Echo is totally awesome and works really well. It works with all different fonts and font sizes.

The other major SLR feature is something called "Mouse Click Feedback." This addresses another common problem in Citrix environments with high latency, namely, the fact that users click on a button, become impatient, and click on a button again before the first click registered. Then, when the application's response finally makes its way to the user, whatever the user clicked on comes up twice. Mouse Click Feedback works by adding the hourglass symbol to the arrow cursor the instant the user clicks the mouse anywhere within the boundaries of the remote ICA application. It does not *technically* prevent the user from clicking the button twice, but the general idea is that the user will see the hourglass and then have the patience to not click the button again.

In most environments with latency, people use both the Local Text Echo and Mouse Click Feedback components of Citrix's SpeedScreen Latency Reduction. It's important to note that enabling SLR will *increase* the amount of bandwidth a session needs, sometimes by as much as 20%. This is due to the fact that extra data will have to move back and forth to coordinate the text rendering. However, SLR will let the user experience a smoother and more responsive session.

This creates an interesting challenge for you as an architect. In order to lower latency you need free up bandwidth by decreasing the ICA data transfers. In order to overcome high latency, you need to enable SLR which increases bandwidth consumption. Which works best for you? That's why you have a test lab.

Packet Loss

If you lose enough packets, your session will eventually disconnect. From a percentage standpoint, both the ICA and RDP protocols can handle even a 10% packet loss rate, and anything higher than that means that you have a major networking issue to deal with.

Refer to the section at the end of this article about troubleshooting dropped and disconnected sessions for more information.

Out of Sequence Packets

Like lost packets, ICA and RDP can easily deal with the "standard" amount of packet sequence issues. Any more significant issues will affect the flow of ack and syn packets and will be a major problem, certainly much bigger than any Citrix or Terminal Server performance issue.

Jitter

There's also not too much you can do about jitter other than minimizing your ICA traffic size, forcing SpeedScreen to be used, and crossing your fingers.

MTU Size

If you've determined that the MTU size of some component of your network is smaller than 1500, you can experiment with a smaller MTU size on your Citrix or Terminal Server. I use the word "experiment" here because you would have to work with a network sniffer to see how the changes you make on the server affect the actual number of packets that go across the network.

Windows is smart enough to automatically configure the MTU size for each network adapter based on what that adapter requests. However, you can override the default settings by forcing an adapter to use a smaller size. This is done in the registry on an adapter-by-adapter basis.

Key: HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
\Interfaces\adapter ID
Value: MTU

Data: The MTU size in bytes. If you want to enter it as a decimal number be sure to select the "Decimal" button

The only drawback to manually adjusting MTU size is that it applies to all TCP/IP communication for a given network adapter. If you have users connecting wirelessly with an MTU size of 460 and regular users with an MTU size of 1500 then whatever you configure will affect them both.

Is there a happy medium where packets can be evenly split for wireless users but not too small to be inefficient for regular users? That's something that you'll need to figure out with your test environment and a network sniffer.

(By the way, the “wireless” users we refer to here with small MTU sizes are mobile wireless users connecting via mobile phone networks. 802.11 wireless connections use the standard MTU size of 1500 bytes.)

Troubleshooting Random Disconnects

The last thing we need to look at in this article is something that we’ve pretty much avoided so far: dealing with random session disconnects and dropped sessions. We’ve spent a lot of time discussing the various performance issues of sessions when they are actually connected, but what happens when a user is happily working and then suddenly their session is disconnected?

Rather than going through all the step-by-step methodology of how to troubleshoot (check the cable, try to ping the server, etc, etc.), we’ll focus here on what happens that causes the ICA or RDP protocol to disconnect a session.

A disconnect is actually a complex procedure. Several things need to happen for a session to become disconnected. (It’s a lot more involved than just yanking the network cable out of the wall.)

1. The client needs to send packets to the server that go unanswered. After a period of time the client has to stop trying and give the user an error message indicating that communication with the server has ended.
2. The server needs to realize that the client device is no longer sending packets. Then it needs reclassify the user’s session as “disconnected.” (Or it has to reset the session if disconnected sessions are not allowed.)

Let’s take a more detailed look at each of these, beginning with the client.

What happens when a session is dropped—a client’s perspective

Imagine an ICA or RDP client is being used normally. The client software converts the keystrokes and mouse movements into ICA or RDP data packets and then hands them off to the TCP/IP interface. The TCP/IP and networking subsystems handle the complexities of adding sequence numbers, sending the data, and receiving ack packets from the server. If for some reason the client doesn’t receive the ack packet after the retry timer is up, you’ll remember it doubles the time and tries again, then doubles it again and tries again, etc. All of these retries handled by the low-level TCP/IP subsystem—they’re hidden from the application itself. However, once the max retry count is reached (remember this defaults to five in Windows), the TCP/IP subsystem reports back to the application that the packet could not be sent.

What happens next? That depends on the application. Microsoft’s remote desktop connection client, for example, causes the remote RDP window to fade to grayscale. Then, a little box pops up on the screen indicating that the system is now trying to reconnect with attempt 1 of 20. The client software sends another packet down to the TCP/IP subsystem. The TCP/IP subsystem will transmit that packet, wait for the ack, double the timer, retransmit, wait, double the timer, etc. up until the five retry counts have been hit. It will then report back to the application that the packet

transmission failed, and the RDP client will update the window to say “attempt 2 or 20” and the whole process is repeated.

If the remote server finally does respond (by sending data or an ack) then the TCP/IP subsystem notifies the application, and the RDP client springs back to life in full color.

What happens when a session is dropped—a server’s perspective

The server also needs to keep track of which sessions are active. After all, if one user disappears then the server will need to change the status of that user’s session from “active” to “disconnected.”

Remember that we discussed that the ICA and RDP protocols are very bursty. This means that they are almost dormant in-between bursts. This can cause a problem if a network connection fails during one of these quiet periods since the server would not even know the client has become disconnected since it wasn’t expecting any client input. (Of course meanwhile the client may be stepping through all of its reconnection steps but the server wouldn’t know this since the connection has been lost.)

Why does it matter if the server knows whether or not a client is disconnected? In MetaFrame XP, ICA clients cannot connect to active sessions; they can only connect to disconnected ones. The problem this causes is that if a client drops the connection and then quickly tries to connect again, the server won’t realize that the first connection was dropped, leaving the session in an active state. Upon reconnection, the server will not be able to assign the active session to the user, so the user will have to create a new session from scratch. By the time the server realizes the first connection is lost it’s too late—the user already has a second session.

(Citrix has addressed this from the client side with the “auto client reconnect” functionality of newer ICA clients. Auto client reconnect first checks to see if the session that was dropped is still active, and if so, it disconnects it before connecting back to it.)

To address this problem from the server side, a feature called “ICA Keep-Alives” is used. ICA Keep-Alives are enabled by default in all new MetaFrame environments.

ICA Keep-Alives use a timer to track how long it’s been since they last had contact with an ICA client. Once the timer is up, the server sends a Keep Alive packet (kind of like an “ICA ping”) to make sure the client is still there and running. If not then it switches the status of the session to “disconnected.”

ICA Keep-Alives are another one of those things that you have to balance. For example, if you set your Keep-Alive interval to be five seconds then your environment will be very responsive and dropped sessions will be identified quickly. However, a five-second keep-alive interval also means that a lot of unnecessary traffic will be flowing across your network as your server checks in with each session every five seconds.

There's one other interesting aspect of the ICA keep-alives timer we should cover. To see where we're going here, think back to how an application interfaces with the low-level TCP/IP stack and the role of the TCP retry count.

Let's assume that you set your ICA keep-alive interval for 30 seconds. After 30 seconds the server constructs an ICA keep-alive packet and hands it off to the TCP/IP interface to be sent to the client device. If the client device really has been lost, then of course it won't send back an ack packet. After a certain amount of time (remember this is dynamic based on the current session parameters) the server will try again. Let's say the default timer is two seconds, meaning now 32 seconds have gone by since the client last checked in. If your TCP Max Retry count is set to 5 (the default), then your server will also send out that ICA packet at 36, 44, and 60 seconds. Only after the 5th failure at 60 seconds will the TCP/IP subsystem report back to the application that the remote computer is not there.

Of course this is just an example, but the dynamic retry intervals could mean that the whole ICA keep alive interval could be two minutes or more on connections that initially have a lot of latency.

How does Session Reliability factor in to this?

One of the new features of Citrix MetaFrame Presentation Server 3.0 Advanced or Enterprise editions is something Citrix calls "session reliability." On the surface, Session Reliability hides the fact that client packets are going unanswered by the server (apart from a small spinning hourglass cursor). The idea here is that by hiding this from the user, the user will have a better experience during small network "hiccups."

Session Reliability is enabled by default in MPS 3 environments when Win32 clients connect with version 8 or higher. When enabled, it wraps the standard ICA port 1494 communication in a new CGP-based protocol that uses port 2598 and sends it off to the server. The Citrix XTE service running on the server receives this traffic, peels off the reliability layer, and passes it internally to port 1494.

From a dropped session troubleshooting standpoint, the important thing to know about Session Reliability is that it does not use ICA keep-alives since the CGP protocol wrapper contains its own mechanisms for this.

In the real world, your servers will probably have a mixture of standard connections and Session Reliability connections (especially since Session Reliability only works with certain clients and never works through Citrix Secure Gateway). This means that you'll still have to work with ICA keep-alives even in newer environments.