TABLE OF CONTENTS

BACKGROUND 4
THE PROBLEM 5
INVESTIGATION 6
1. VALIDATE BASIC NETWORK STABILITY 6
2. VALIDATE THE WIREGUARD TUNNEL 6
3. OBSERVE RDP TRANSPORT BEHAVIOR 7
4. CORRELATE SESSION DROPS WITH SYSTEM METRICS 8
ROOT CAUSE 8
COMMON MISLEADING ASSUMPTIONS 9
RECOMMENDED SOLUTIONS 10
1. INCREASE SERVER RESOURCES 10
2. REDUCE THE GRAPHICAL WORKLOAD 10
A PRACTICAL STARTING POINT FOR RDS SIZING 10
REALITY CHECK 12
KEY TAKEAWAYS 13
CONCLUSION 13
RELATED ARTICLES 14
TAGS 14

WHEN IT’S NOT THE NETWORK: AN RDP INVESTIGATION THAT LED ELSEWHERE

Author(s): Louis Ouellet

In a recent deployment, I was asked to investigate unstable Remote Desktop (RDP) sessions to a remote server accessed over a site-to-site VPN.

At first, the explanation sounded simple: the VPN was unstable. Users were being disconnected, the target server was remote, and all the symptoms seemed to point in the same direction.

On paper, the environment was straightforward:

- Local network: `192.168.115.0/24`
- Remote network: `192.168.201.0/24`
- Original VPN: IPsec
- Temporary replacement VPN: WireGuard
- Target server: `192.168.201.100`

- Multiple users connecting via RDP

But as is often the case in infrastructure work, the first explanation turned out to be the most convenient one — not the most accurate.

> This is one of those cases where everything looks like a network issue — until you start proving what is, and is not, actually failing.

BACKGROUND

The issue did not begin with WireGuard.

Originally, connectivity between both environments relied on an IPsec tunnel. When that tunnel stopped working, the first claim was that the tunnel still appeared up on the remote side, so the problem must be somewhere on my end.

From my side, the evidence told a different story. Packet captures on the WAN interface showed my firewall sending traffic out, but receiving no replies. A traceroute went one step further: the remote IPsec endpoint could not even be reached.

At the same time, users had already been reporting lag when connecting to `192.168.201.100`. Those complaints had existed before the IPsec outage, which made the situation more complicated. There was now a broken tunnel, but there was also a longer-standing performance problem affecting the same remote server.

Even before the deeper analysis, there were early signals pointing toward a potential capacity issue. During conversations with the remote team, it was mentioned that CPU and memory usage were typically around 80% under normal conditions. Based on my experience with RDS and virtual desktop environments, that left very little headroom for peak user activity.

This did not prove anything on its own, but it made resource saturation a plausible hypothesis worth validating.

Before the IPsec tunnel failed completely, I had already shared a few recommendations that could help reduce the load on the remote side, such as publishing the application as a RemoteApp or reducing the graphical burden placed on the server. Those suggestions did not

move things forward, largely because the experience was reportedly different for other users.

Once the IPsec tunnel was no longer usable, the proposed next step was to move to WireGuard.

Deploying WireGuard directly on an RDS server was suggested, but that was not a direction I was willing to take. Running VPN termination directly on a production session host creates unnecessary security and network design problems. Since upgrading pfSense to support WireGuard properly would have required maintenance downtime, I instead built a dedicated Ubuntu Server 24.04 LTS virtual machine to act as a temporary WireGuard gateway and added the required routing on pfSense.

That is where the real investigation began.

THE PROBLEM

Once the WireGuard path was in place, users were able to connect again, but the core issue remained.

They experienced:

- Random RDP disconnections
- Noticeable instability during active use
- Sessions partially reconnecting, with redirected drives and printers coming back on their own
- No obvious packet loss during basic connectivity tests

This created a classic and very misleading situation:

> Ping was stable, but the application was not.

That is exactly the kind of symptom that can send an investigation in the wrong direction if you stop at the first assumption.

INVESTIGATION

Rather than treating the VPN as guilty by default, I worked through the problem one layer at a time.

---

1. VALIDATE BASIC NETWORK STABILITY

The first step was to verify whether the path itself was unstable.

I started with ICMP testing and MTU validation:

```
ping 192.168.201.100 -f -l 1300
```

Through iterative testing, I found that an MTU around:

- `1380 bytes`

worked without fragmentation.

Basic network tests consistently showed:

- No packet loss
- Stable latency

> Conclusion: the path itself was stable enough to carry traffic reliably.

---

2. VALIDATE THE WIREGUARD TUNNEL

The next step was to verify the VPN gateway itself.

I checked the tunnel using:

```
wg show
conntrack -L
iptables -L
```

What I found:

- Stable and recurring handshakes
- No conntrack saturation
- No firewall drops explaining the behavior
- Bidirectional traffic flowing properly through the gateway

> Conclusion: the WireGuard tunnel was functioning correctly and was not dropping outright.

---

3. OBSERVE RDP TRANSPORT BEHAVIOR

Since the basic path and the VPN both appeared healthy, I moved one layer higher and watched the RDP traffic itself using `tcpdump`.

The captures showed:

- Heavy UDP usage
- Bursty traffic during interaction
- Packet sizes commonly in the ~1000 to 1200 byte range
- TCP fallback behavior that was less stable than expected

Example:

```
IP 192.168.201.20.51020 > 192.168.201.100.3389: UDP, length 1237
IP 192.168.201.20.51020 > 192.168.201.100.3389: UDP, length 1005
```

At this stage, it still looked like the issue might be transport-related. The traffic pattern was clearly active, and RDP was making heavy use of UDP, especially during user interaction.

> It still looked like a network problem — but the evidence was no longer lining up with packet loss.

4. CORRELATE SESSION DROPS WITH SYSTEM METRICS

The turning point came when I stopped looking only at packets and started correlating the disconnects with server performance.

To do that, I monitored CPU and memory on `192.168.201.100`:

```
typeperf "\Processor(_Total)\% Processor Time" "\Memory\% Committed Bytes In Use"
```

The result was clear:

- CPU usage repeatedly spiked to `100%` at the exact moment the RDP session became unstable or dropped

> That was the moment the investigation changed direction.

In hindsight, the performance data aligned with the earlier observations: the server had very little margin left once user activity increased.

At that point, the issue was no longer a question of whether packets were crossing the VPN. They were. The real question was whether the remote server could keep up with the workload being placed on it.

ROOT CAUSE

The primary issue was not the VPN. It was server-side resource saturation, especially CPU exhaustion on `192.168.201.100`.

RDP sessions, particularly when users are actively working, generate a surprising amount of graphical and session-processing overhead. In this case, the disconnects were not random at all. They lined up with moments where the server was running out of headroom.

That explains why the symptoms were so misleading:

- The tunnel stayed up
- Ping continued to work
- Packets continued to flow
- But the user session still became unstable

When CPU hits 100% on the RDP host:

- Session processing is delayed
- Graphical updates fall behind
- Transport responsiveness degrades
- The client starts behaving as though connectivity is failing

> What looked like a network issue was, in reality, a compute bottleneck presenting itself as a network symptom.

COMMON MISLEADING ASSUMPTIONS

Several possible causes had to be tested and ruled out along the way:

- A failing VPN tunnel
- MTU or fragmentation problems
- UDP versus TCP transport behavior
- Firewall or conntrack limitations

Those were all reasonable hypotheses. They just were not the main cause of the instability.

> A stable ping does not prove an application is healthy, and an unstable application does not automatically mean the network is at fault.

RECOMMENDED SOLUTIONS

Once the root cause was clearer, the solutions were fairly straightforward.

1. INCREASE SERVER RESOURCES

The most direct fix would be to give the server more headroom:

- Add CPU capacity
- Ensure adequate memory
- Monitor sustained load during user activity

2. REDUCE THE GRAPHICAL WORKLOAD

The second option would be to reduce how much work the server has to do per session.

Publishing the application as a RemoteApp would help by:

- Reducing the amount of desktop rendering required
- Lowering the graphical overhead of full remote sessions
- Improving scalability for multiple users

Other possible mitigations include:

- Reducing display resolution
- Limiting monitor usage
- Disabling unnecessary visual effects

A PRACTICAL STARTING POINT FOR RDS SIZING

One of the lessons from this case is that session host sizing should be based on actual user

workload, not just on whether users can technically log in.

Microsoft’s guidance makes an important distinction between **light**, **medium**, **heavy**, and **power** workloads. In practice, that means the right size for a session host depends less on the number of users alone and more on what those users are actually doing throughout the day.

The table below is a simplified way to think about common user profiles in real environments.

User profile	Typical behavior	Typical apps / activity	Relative load
Light	Limited multitasking, mostly repetitive tasks	Data entry, line-of-business apps, command-line tools, static web pages	Low
Medium	Moderate multitasking, office productivity	Word, Outlook, web apps, PDF viewing, multiple browser tabs	Moderate
Heavy	Frequent multitasking with communication and dynamic apps	Outlook, Teams, Zoom, Office apps, many browser tabs, remote file access	High
Power	Demanding visual or compute-heavy workflows	Development tools, content creation, CAD, video editing, high-resolution multi-monitor use	Very high

For many small business environments, users often fall somewhere between `medium` and `heavy` rather than at the `light` end of the scale, especially when multitasking, video calls, and browser-based business apps are part of the daily workflow.

As a practical starting point, the following sizing guidance is often more realistic for multi-session RDS deployments:

User profile	Suggested starting point	Approximate user density	Notes
Light	8 vCPU / 16 GB RAM	Up to 6 users per vCPU	Suitable for basic task workers with limited multitasking

User profile	Suggested starting point	Approximate user density	Notes
Medium	8 vCPU / 16 GB RAM	Up to 4 users per vCPU	Better for office productivity and moderate multitasking
Heavy	8 vCPU / 16 GB RAM minimum, often more in practice	Up to 2 users per vCPU	More appropriate for users running communication tools, many apps, and active multitasking
Power	16+ vCPU / 56 GB+ RAM, sometimes GPU-backed	Around 1 user per vCPU or less	Best for graphics-intensive, high-resolution, or compute-heavy sessions

These figures should be treated as a starting point only. Real capacity depends on user behavior, login patterns, storage performance, network conditions, and how much headroom is left for short bursts of activity.

In other words, if a session host is already sitting near 80% utilization under normal conditions, it likely has very little tolerance for peak demand.

REALITY CHECK

From a troubleshooting perspective, the investigation led to a fairly clear conclusion: the bottleneck was on the remote server.

The two practical responses were:

- Increase available resources on `192.168.201.100`
- Reduce the workload being pushed through full RDP sessions

However, that was not the direction ultimately taken.

Instead, the chosen response was to deploy additional client devices and establish separate tunnels for individual users.

That distinction matters, because it highlights a common disconnect between diagnosis and implementation:

- The investigation pointed to a compute problem
- The chosen response focused on network topology

> More tunnels do not solve a server-side resource bottleneck.

That does not make the investigation any less valuable. If anything, it reinforces an important reality of infrastructure work: finding the root cause and getting the root cause addressed are not always the same thing.

KEY TAKEAWAYS

- Do not assume the network is the problem just because the service is remote
- Validate each layer independently before moving to the next
- A healthy VPN tunnel does not guarantee a healthy RDP session
- Application instability can be caused by server-side resource exhaustion even when connectivity appears normal
- Root cause analysis is only useful if the chosen solution matches the layer where the problem actually lives

---

CONCLUSION

This case was a useful reminder that infrastructure problems are often shaped as much by assumptions as they are by technology.

What started as an investigation into an “unstable VPN” turned into a deeper look at session behavior, packet flow, transport patterns, and finally server-side performance. By the end of the process, the path itself had been validated, the temporary WireGuard gateway had been proven functional, and the real bottleneck had been identified elsewhere.

Not every investigation ends with the solution you would choose. But there is still value in doing the work properly, documenting the evidence, and understanding where the problem truly resides.

RELATED ARTICLES

- Fixing CrowdStrike Issue on Windows
- Git --- Let's talk about it
- How to Check a Hard Drive Health from the Command-Line
- How to Connect an iSCSI Target on Windows Server 2022
- How to disable Telemetry on Windows 11

All Related Articles

TAGS

general, blog, rdp, network, wireguard, investigation, security, windows, linux
View the discussion thread.

From:
https://laswitchtech.com/ - **LaswitchTech**

Permanent link:
https://laswitchtech.com/en/blog/2026/03/22/when-it-s-not-the-network-an-rdp-investigation-that-led-elsewhere

Last update: **2026/03/23 18:25**