Blog Draft:
How Subspace Supports Real-Time Application Development Using TCP and UDP Protocols

Project Information

Word count:	~1200
Objective:	In this post, Subspace will announce that the Subspace platform now supports both TCP and UDP channels. We will describe how Subspace optimizes the messages sent to and from clients. And we'll dive deep into the ways in which Subspace can be customized for specific real-time applications.
Target audience	Executives, engineers, and team members at small-to-large game publishers/studios and ISP/telcos. Real-time application developers rely on both protocols.
Visuals:	*Subspace must provide all required visual assets.*
Target keywords:	TCP UDP network support, real-time optimization for TCP & UDP

Headline:

How Subspace supports real-time application development using TCP and UDP protocols

TL;DR: Subspace offers max-performance support for both TCP and UDP protocols, enabling today's real-time applications to reach the pinnacle of application enablement, speed, reach, and reliability.

Read time: *5 mins*

URL: /blog/real-time-support-tcp-udp

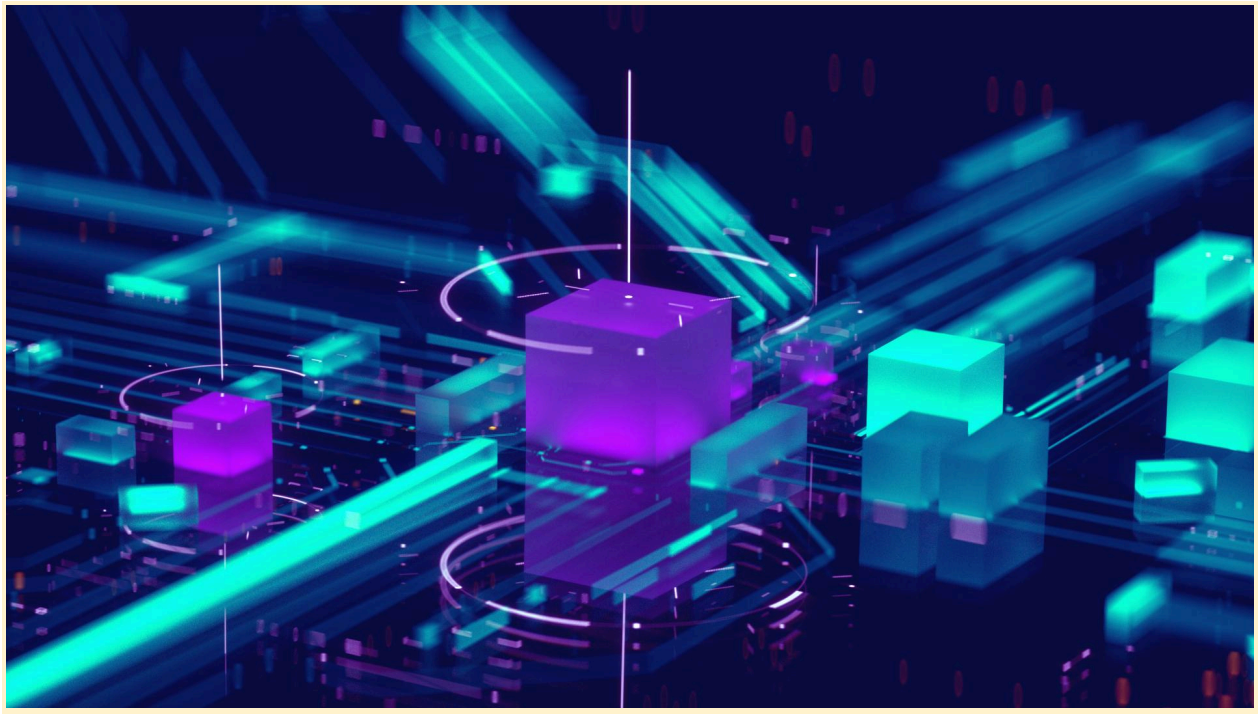
Meta-description: Real-time internet requires optimized TCP & UDP protocols. Read more about how Subspace accelerates performance on both.

Hacker News Social Post: Optimized support for TCP & UDP protocols is possible for real-time applications.

Internal Sharing Summary:
Hi team, we have a great post highlighting Subspace's unique ability to support and accelerate real-time applications on both TCP and UDP protocols. We'd greatly appreciate it if you shared it with your network.

Thank you!

Header image recommendation:
This image (overlay removed)



Draft:

How Subspace supports real-time application development using TCP and UDP protocols

Bottom line: Internet applications rely on both TCP and UDP protocols. Finally, one network provider—Subspace—offers max-performance support for both protocols, enabling today's real-time applications to reach the pinnacle of application enablement, speed, reach, and reliability.

TCP and UDP

Two IP-based protocols, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), work within the internet's TCP/IP protocol stack. They govern how packets get sent between network nodes. To radically oversimplify the matter, TCP uses a three-way handshake to establish an inter-node connection. The dialogue boils down to:

Node 1: Hi there.
Node 2: Hi there. I hear you.
Node 1: I hear you, too.

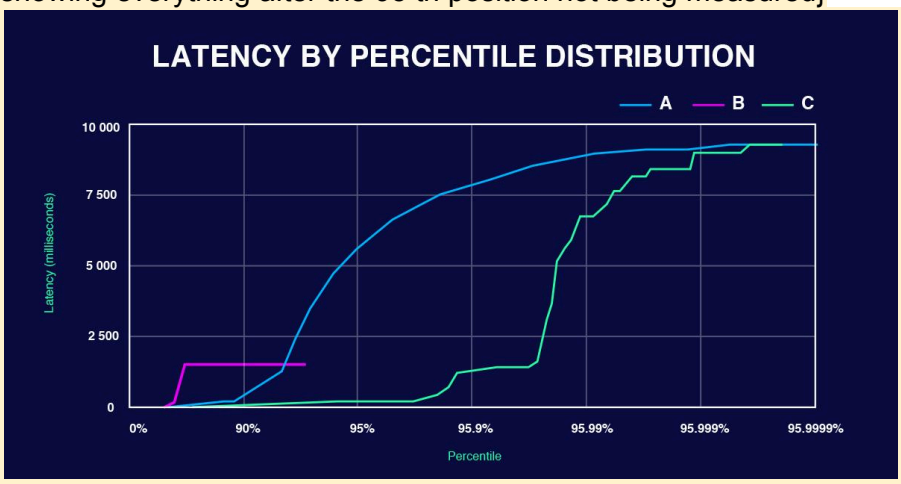
With this witty repartee completed, Node 1 can begin sending data packets to Node 2. TCP's imperative is to never lose a packet, no matter how long it takes to deliver. For example, if Node 1 sends eight packets, but the fifth packet is not received, Node 2 will ask Node 1 to retransmit the fifth. Once all eight packets have been received, TCP makes sure they are in the proper order, after which they are passed up the software stack. TCP is ideal when data loss is intolerable, as with email or Web content.

UDP offers neither handshaking nor error correction/packet recovery (although applications may take over this task). Naturally, UDP is faster and creates less traffic than TCP. Gaming and video are suited to UDP because humans won't mind a stray hiccup or dropped frame on rare occasions, but we experience stream-rage when the entire experience stops to wait for slow packets. UDP looks stellar when you benchmark it on a LAN, but expect to lose 10% to 50% of your packets across many stretches of the public internet.

Ready for Real-Time

Subspace launched its next-generation network and software stack for UDP. The gaming market, in particular, was desperate for a faster, more reliable alternative to the global public internet, and Subspace worked out how to solve UDP packet loss issues without sacrificing protocol speed.

{Subspace to provide graphic depicting 95th percent measurement - something like this - but showing everything after the 95 th position not being measured}



Now, Subspace is bringing the same class of protocol support and improvement to TCP. Internet providers struggle with TCP for numerous reasons. The foremost of which may be head-of-line (HOL) blocking, meaning that if the first packet, or any packet for that matter, doesn't arrive-- it holds up the entire packet line. HOL blocking is often caused by network links experiencing issues, such as congestion or disruption. An ISP might only have 1% or 2% of its traffic experiencing such problems. That can clobber an application's real performance, but because most ISPs don't report or even measure traffic beyond the 95th percentile, the issue doesn't get recognized.

Because Subspace explores the quality of *every* traffic link and dynamically optimizes them in real-time—using proprietary methods and algorithms that would otherwise require over 10,000 servers per continent to replicate—our TCP performance will show a clear, even startling, improvement over the public internet.

Subspace brings a level of control to the internet that it was never meant to have. The internet was designed to be resilient, not performant, in part because of how the internet's Border Gateway Protocol (BGP) selects a traffic path with the fewest autonomous system hops between endpoints, not the fastest route between them. The control required to put performance first is neither easy to build nor easy to facilitate since routing rules often differ from network to

network. Despite all the progress made from early dial-up to locally-cached CDNs, evidence of how much the public internet struggles to facilitate real-time applications remains.

{Subspace to provide related latency graph - This is the most closely related we have - but we are looking to show an 80% improvement graph.}

Illustrative distribution of latencies via normal Internet routing and via Subspace, 24 hours, demonstrating latency reduction.

Millions of Subspace users see and feel how we reduce latency by as much as 80%, which *exceeds* the performance standards for many real-time applications. In multiplayer games, the industry standard for smooth gameplay is 20 ms of latency or less. In chat applications, the maximum message delivery latency is 250 ms. Even the FCC places access network limits of 35 ms upstream and 15 ms downstream for QoS-enabled services, such as VoIP “home phone.” Access networks being the space between the home network and the aggregation network.

How Subspace is Unique

Subspace is an acceleration network. Specifically, we optimize data activity across all seven layers of the Open Systems Interconnection (OSI) model, from the physical interconnects in our servers to the tweaks we help developers implement for maximum performance in the application layer. Such end-to-end improvements simply aren’t possible for ISPs, which are bound by conventional BGP protocols. Rather, Subspace improvements apply across networks,

such as in connections between Verizon and AT&T. Often, these are the weakest links in the broad internet data flow, but Subspace removes the risk that they will bottleneck performance.

{subspace to provide connectivity graphic - inspiration for branding, another for connection}

Similarly, recall the differences between TCP and UDP. When a developer selects one of these protocols for an application, it's with "either/or" compromises in mind. Can the application withstand HOL blocking lag and potentially seconds of packet retransmission congestion? If so, TCP is best. Can the application survive losing packets like sand through a sieve? If so, you'll want UDP.

Of course, no developer wants to lose speed or information *at all*. And because Subspace supports both protocols, allowing applications to run both over its optimized network, no compromise is necessary. Developers can realize real-time results without data degradation from both protocols.

Normally, when setting up TCP or UDP connections at the socket level, multiple communications happen simultaneously. This can take excessive time when an API issues a simple "open" or "connect" call. These delays impair real-time applications. TCP accepts these

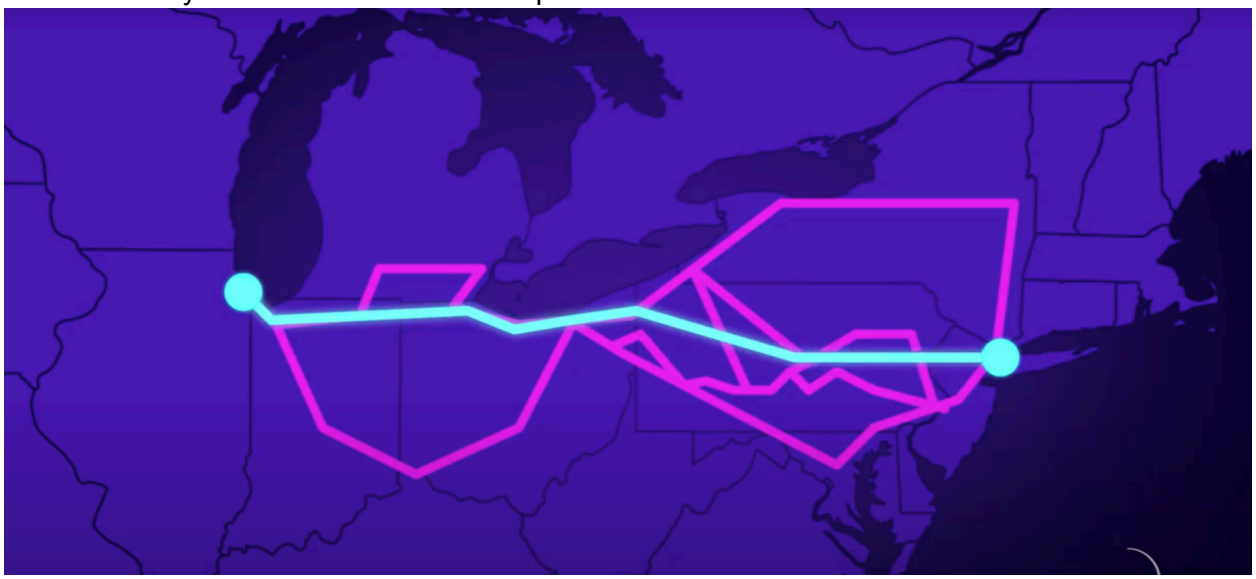
delays as a given, which is why TCP-based chat applications often employ waiting rooms or display the much-loathed “loading” beach ball while users wait to be connected.

Subspace is push-to-talk rather than wait-to-connect. Application performance is such that human responses are the slowest element.

Achieving Protocol Optimization

How does Subspace achieve this? We mentioned optimizing across all seven of the OSI layers. Beyond that, and more specifically, every packet that crosses through Subspace passes through optimization software that:

- Determines and predicts which paths will cause problems and reroutes traffic accordingly
- Optimizes and tracks packets
- Selects the *fastest* path, not the shortest
- Actively measures the final 5% of packets



{Subspace to provide graphic - capture related to the shortest path animation in Bayan video}

The public internet’s flaws and compromises aren’t a secret. We all live with them, every day, in every online application we use. Those flaws persist because fixing them is like rebuilding a single train system across a massive, densely populated city. Many have tried, and no one has succeeded…which is why Subspace decided to create a high-speed monorail above it all. We replaced nothing. We simply built a better solution with junctions that interface seamlessly with the regular traffic below.

Building a monorail able to support everyone isn’t easy, either. But we did it. And we built all that infrastructure and software optimization so you don’t have to.

With Subspace, users can finally have the best of both worlds, with support for both TCP and UDP and greater security, speed, and reliability throughout the network. This is the level of quality the next generation of internet applications can be built on. Developers can build on it today, and the only thing customers will notice is their flawless experiences.

Learn more about precision measurement {link when live}