QUIC Protocol Overview for Enterprises

With Packet Analysis Examples

Bill.Alderson@SecurityInstitute.com

Course PDF https://CE.SecurityInstitute.com/QUIC

ISF 2024

25th Annual

Information Security Forum for Texas Government

Packetman007


QUIC
REPLACES
TCP-SSL-HTTP
READY ?
INNOTECH
ISSA
Information Systems Security Association
International
CAUTION
QUIC
PROTOCOL
25th Annual
ISF
2024
LIVE
STREAM
SharkFest'23 US
Wireshark Developer and User Conference • San Diego, CA • June 10-15

QUIC Transport Protocol
IETF rfc-9000 & related rfcs

Course PDF https://CE.SecurityInstitute.com/QUIC

Windows Server 2025 Overview of what's New

Windows Server Summit 2024 | March 26-28 2024

QUIC Protocol Is Rapidly Replacing TCP!

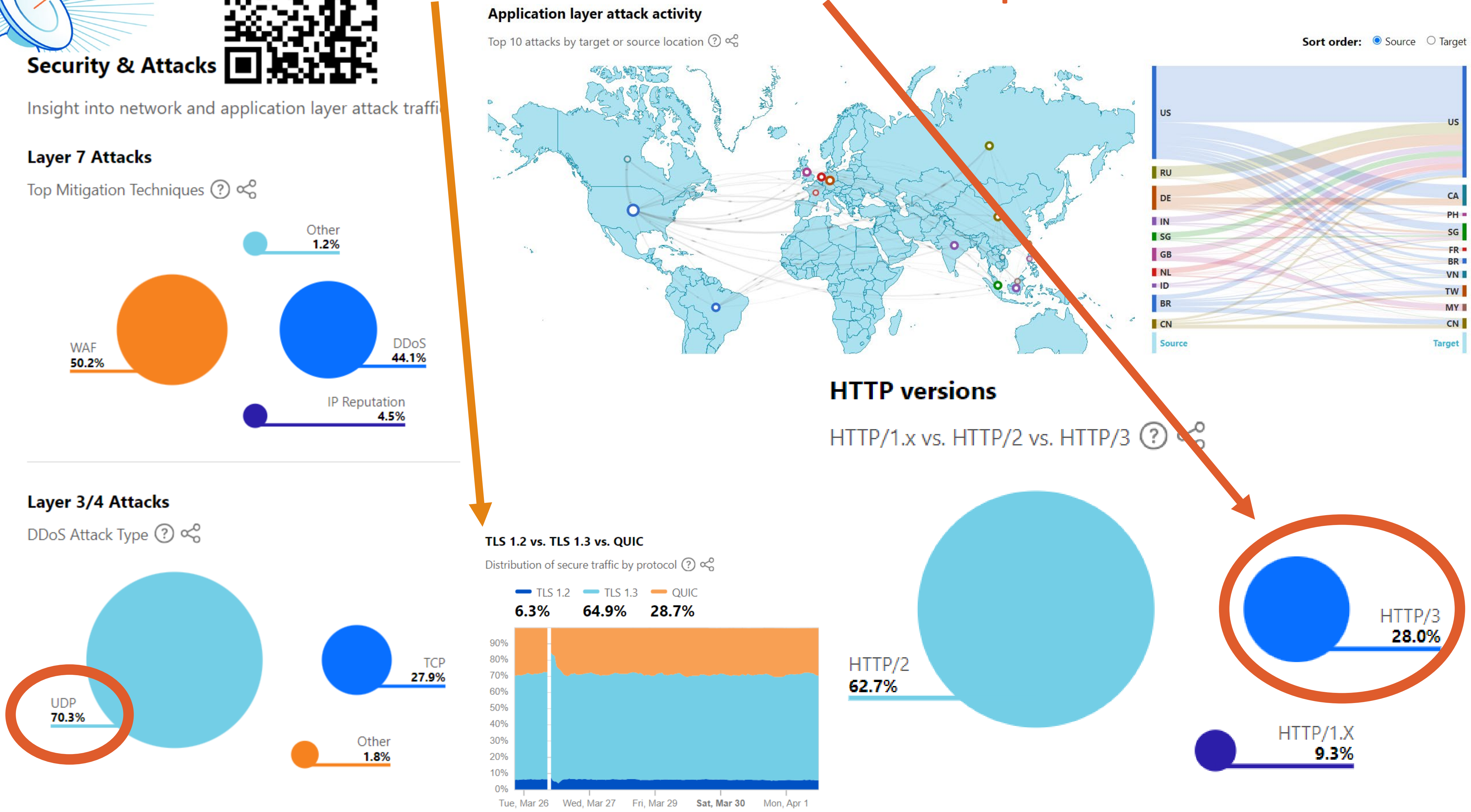
Every Major Client Web Browser
Chrome – Edge – Safari – Mozilla

Most Social Media Apps 90%+
Google – Facebook
Apple – Microsoft Server 2022+
CDN’s Cloudflare - Fastly

SharkFest 2023

- How QUIC Works And What Are The Security Concerns?
- Why QUIC is Faster? Show You How To Perform “Packet Based Analysis”
- Distillation of 1,000+ IETF rfc-9000 and Related Standards Pages
- Why Does It Lower Firewall Sessions 20 to 1 vs TCP?
- How to Implement QUIC on Your App or Site - Fast!

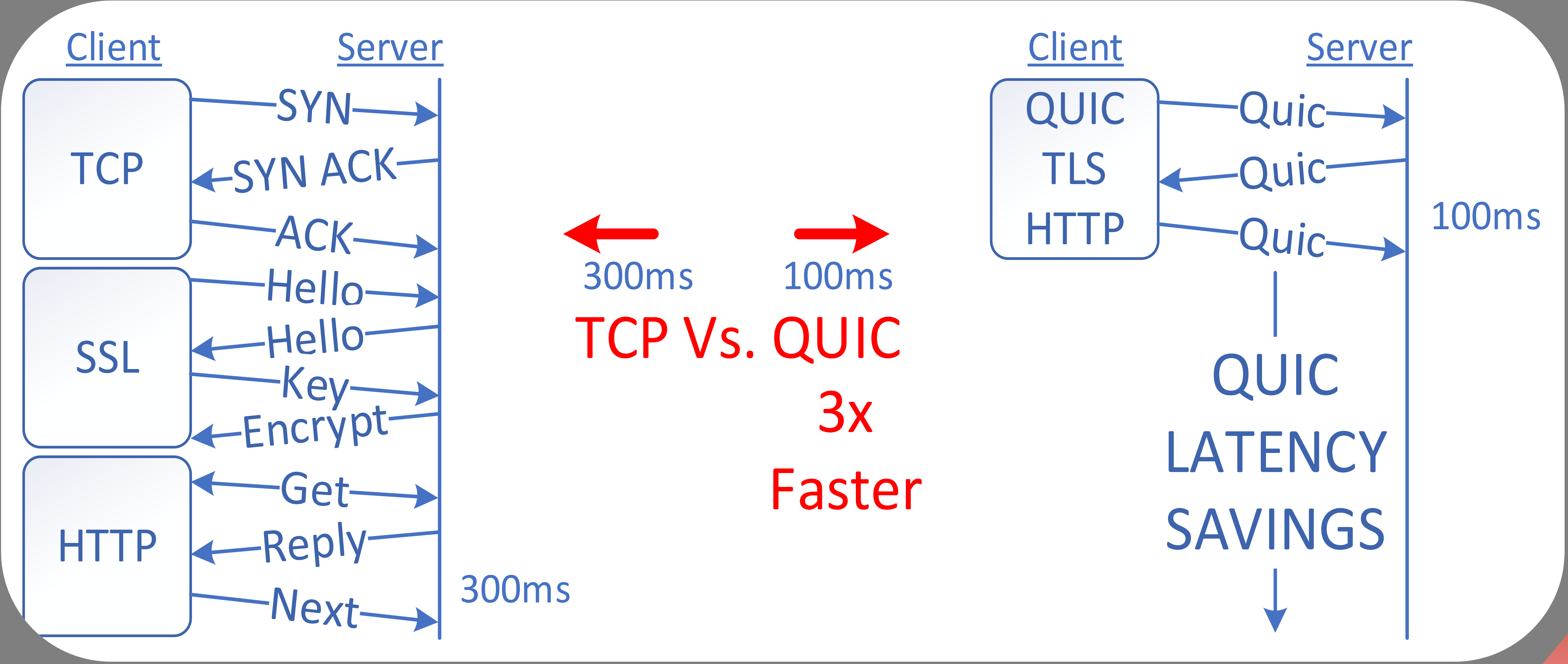
QUIC Protocol Adoption



Current Internet 1x

QUIC 3x Faster

Basics of QUIC Protocol



Firewall Vendors Tell Enterprises "DENY" QUIC

Firewall Vendors that recommend Blocking QUIC UDP Port 443 in Enterprise Networks

Cisco
PaloAlto Networks
CheckPoint
Fortinet

Each vendor provides specific instructions to block QUIC

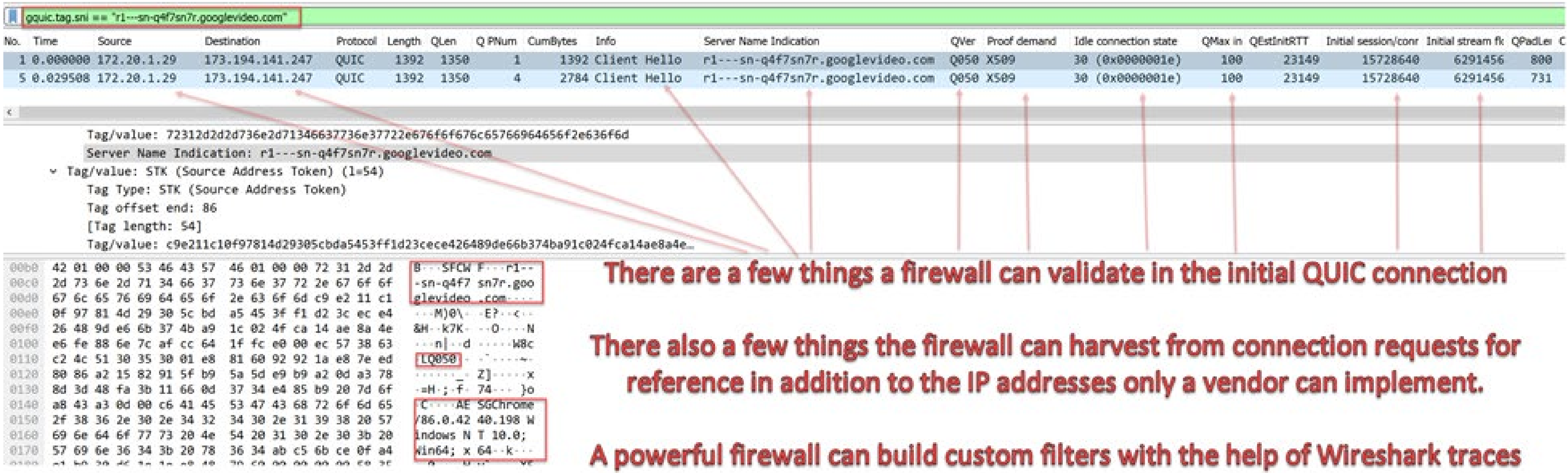
Web Performance at a Price...

An "Attractive Nuisance" Until Enterprise Firewall Improvements Secure Its Use

GotQuic?

- 2012 accidentally discovered... by Jim Roskind at Google, now AWS
- Google, YouTube, Gmail, Facebook, Microsoft Uber and Cloudflare already use QUIC!
- Distant or rural users receive the biggest performance gain.

QUIC Initial Connect – some headers exposed

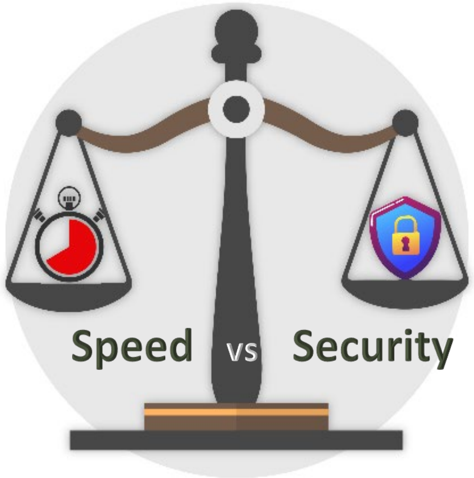


There are a few things a firewall can validate in the initial QUIC connection

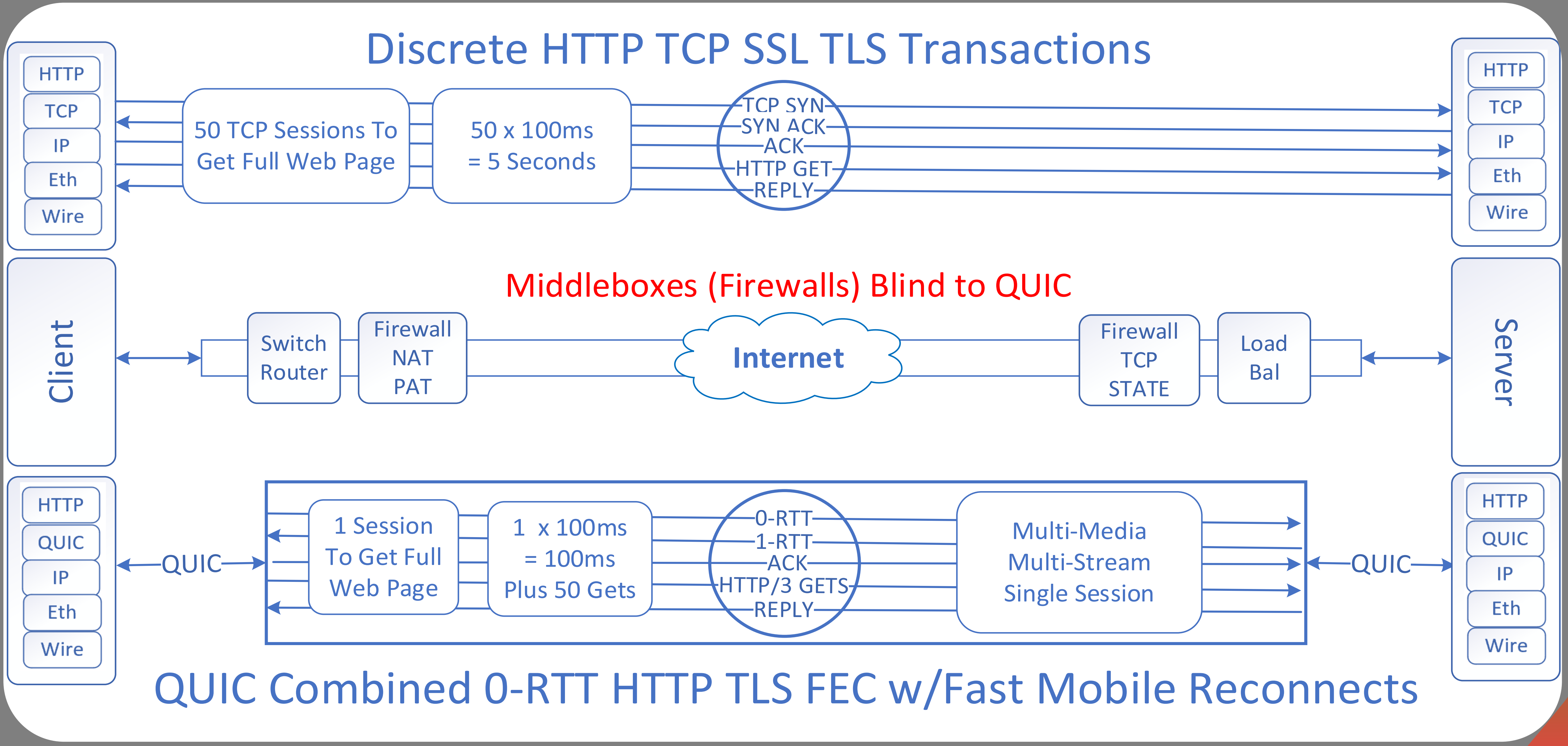
There also a few things the firewall can harvest from connection requests for reference in addition to the IP addresses only a vendor can implement.

A powerful firewall can build custom filters with the help of Wireshark traces

- Enterprise firewall improvements need to identify well formed QUIC packet headers
- Improve and upgrade SMB - Home router-firewalls to identify QUIC
- Technically it is possible upgrades can be accomplished at reasonable costs
- New firewalls worldwide might cost $1 Trillion and take 10+ years

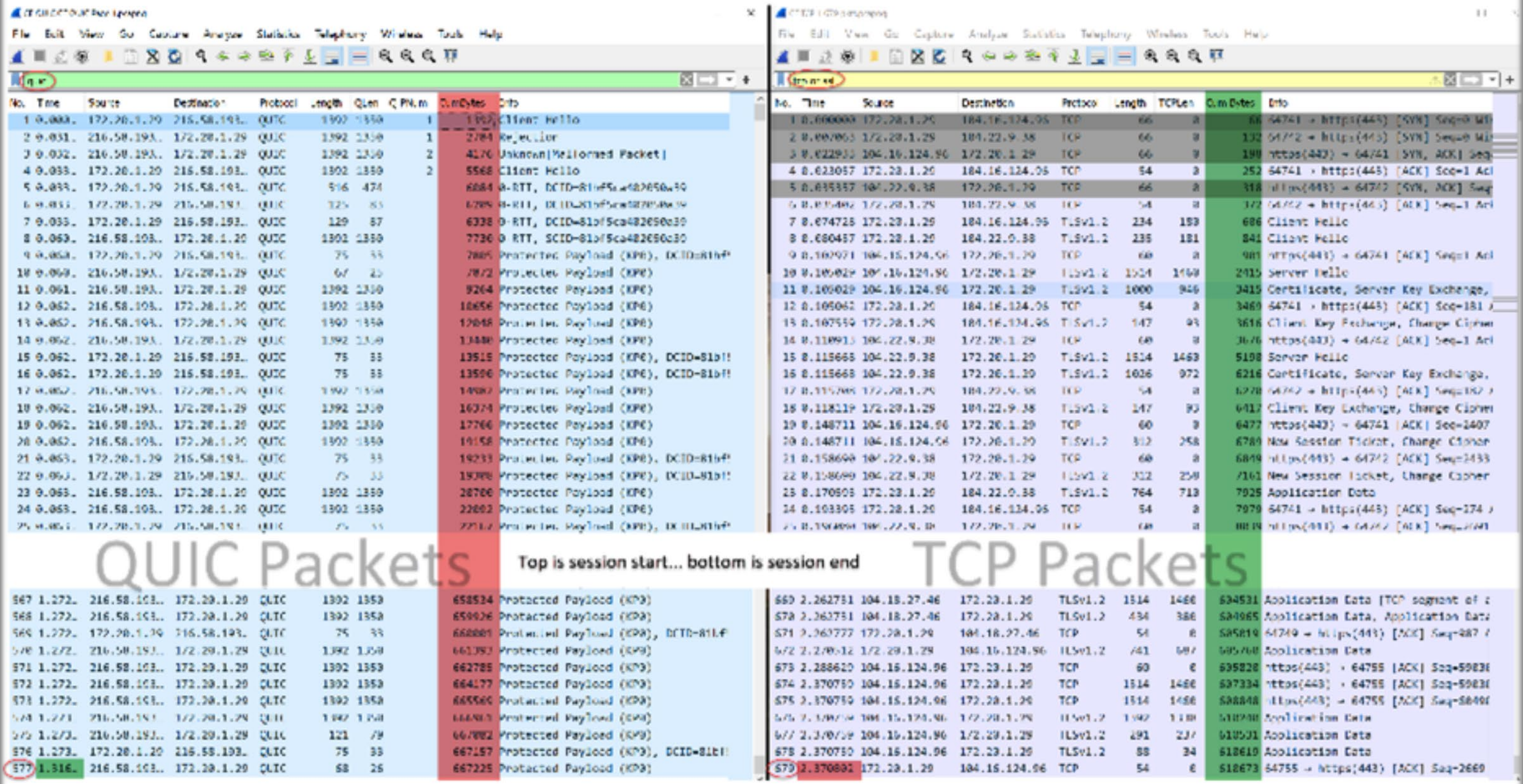


Why Middleboxes (Firewalls) Blind to QUIC?



Benchmarks Prove It Faster!

QUIC vs. TCP Full Page Packet by Packet Performance Analysis



Full Page Load With Multiple Objects

Measurement	TCP	QUIC	QUIC RAW Advantage	Advantage QUIC HTTP/3	Difference
Sessions	13	1	12	92%	Less Sessions
Packets	679	577	102	15%	QUIC Fewer Packets
Time span, seconds	2.371	1.317	1.054	44%	QUIC Significantly Better Time, seconds
Average pps	286	438	152	53%	QUIC Better Average pps
Average packet size	899	1156	257	29%	QUIC Larger packet size
Bytes	610673	667225	-56552	-9%	TCP Fewer Bytes
Average bits/s	2060000	4053000	1993000	97%	QUIC Better Throughput bits/s

Benchmark Page https://cloudflare-quic.com

Benchmark Results

Single Object One Video Load

Measurement	TCP	QUIC	QUIC RAW Advantage	Mixed Results TCP HTTP/2	Difference
Sessions	1	1	0		0% Same Sessions
Packets	1985	2123	-138		-7% TCP Fewer Packets
Time span, seconds	40.332	40.241	0.091		0% QUIC Marginally Better Time
Average pps	49	53	4		7% QUIC Better Average pps
Average packet size	1089	1192	103		9% QUIC Larger packet size
Bytes	2161738	2530541	-368803		-17% TCP Fewer Bytes
Average bits/s	428000	503000	75000		18% QUIC Better Throughput bits/s

Full Page Load With Multiple Objects

Measurement	TCP	QUIC	QUIC RAW Advantage	Advantage QUIC HTTP/3	Difference
Sessions	13	1	12		92% Less Sessions
Packets	679	577	102		15% QUIC Fewer Packets
Time span, seconds	2.371	1.317	1.054		44% QUIC Significantly Better Time, seconds
Average pps	286	438	152		53% QUIC Better Average pps
Average packet size	899	1156	257		29% QUIC Larger packet size
Bytes	610673	667225	-56552		-9% TCP Fewer Bytes
Average bits/s	2060000	4053000	1993000		97% QUIC Better Throughput bits/s

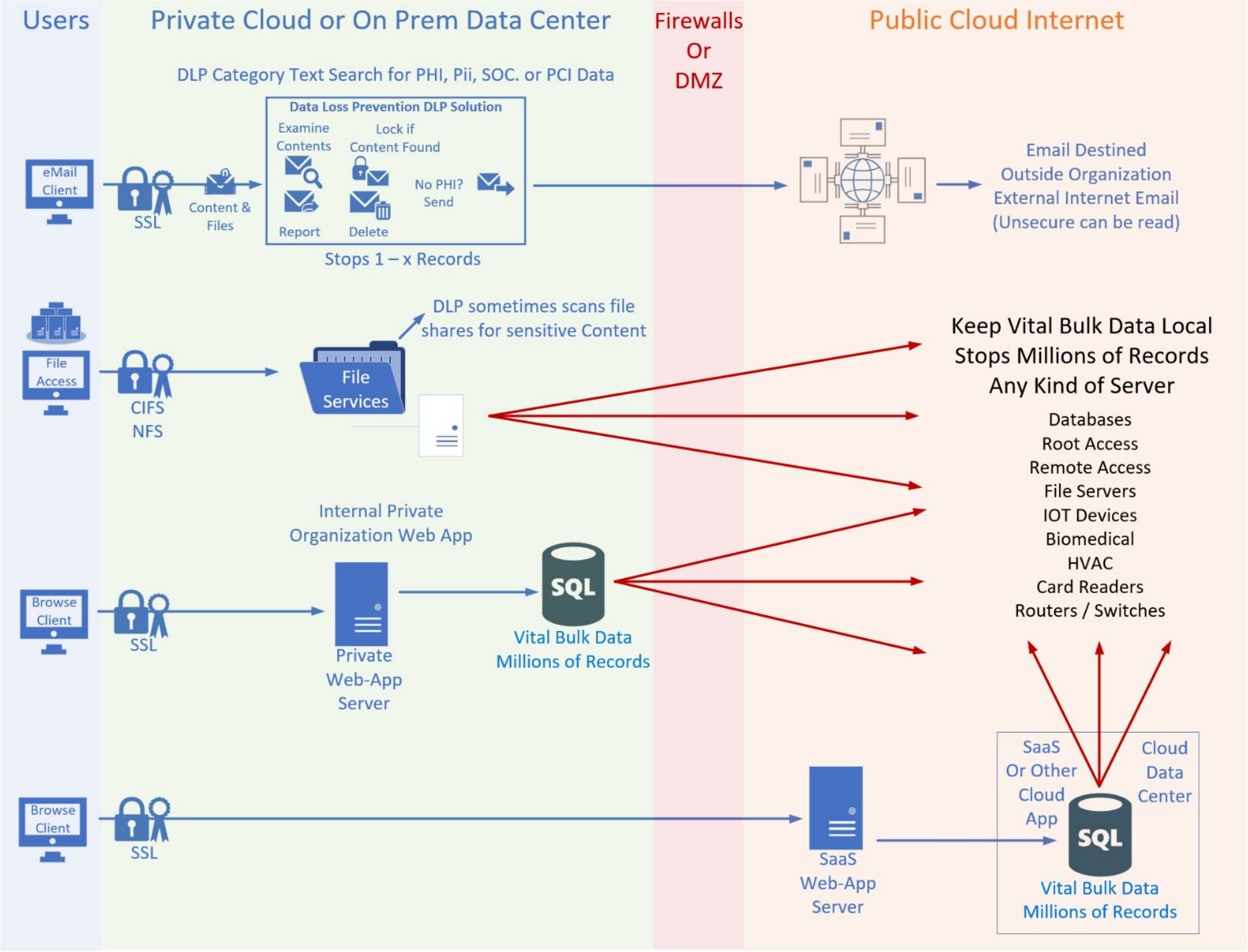
Triple+ Web Performance at a Price...

3x FASTER | SECURITY THREATS

What Will You Do?

QUIC

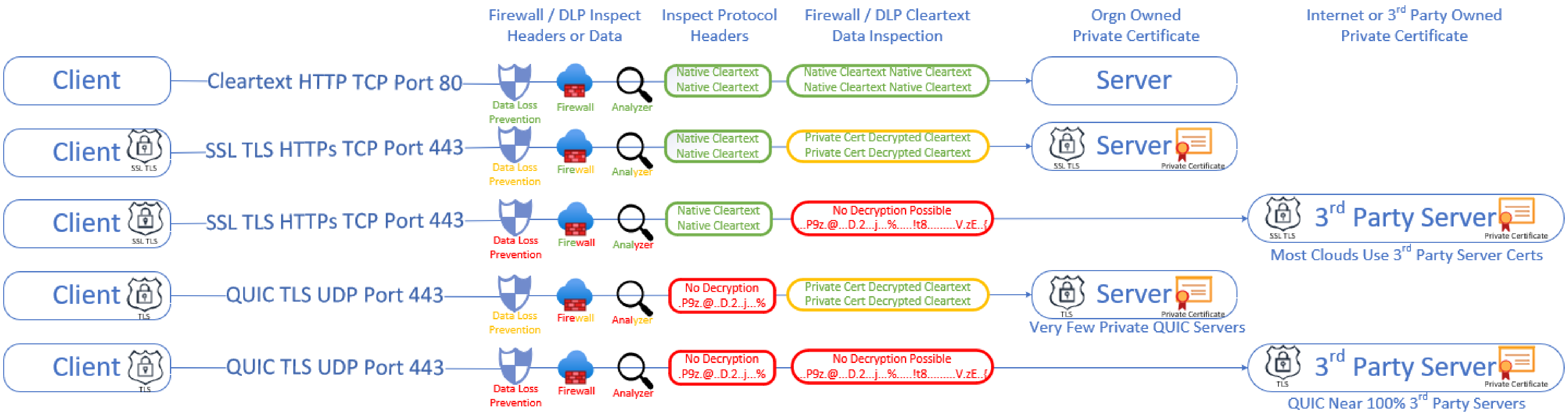
DLP May Not Check QUIC TLS 1.3!!!!!



When does Encryption Prevent DLP Scanning?

Using 3rd party private cert prevents scanning

90+ Percent of Malware arrived Via Encrypted Traffic: WatchGuard

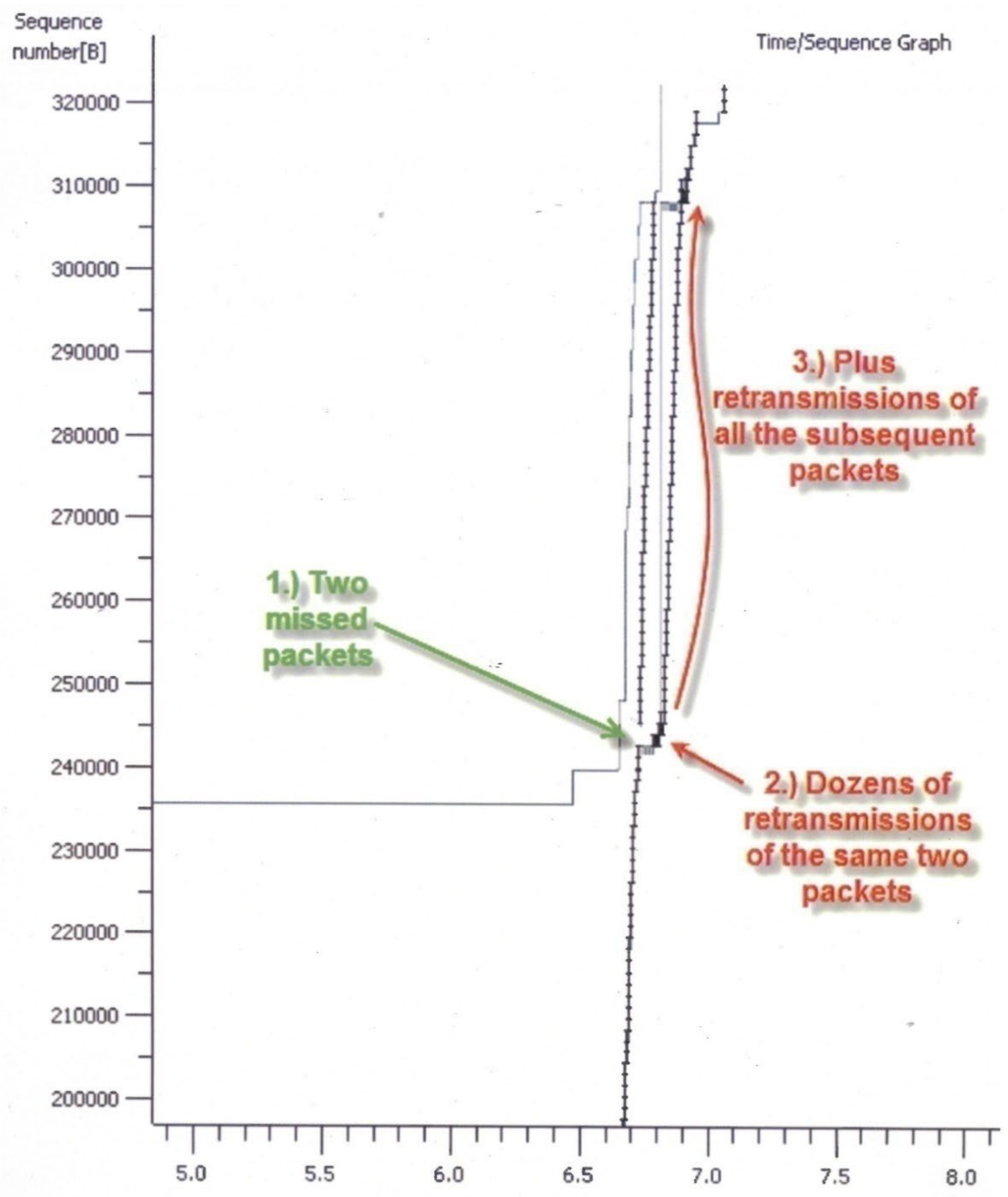


Here is a message from Vinton Cerf, known as the father of the Internet, created to encourage us on Security Zero-Day Prevention.

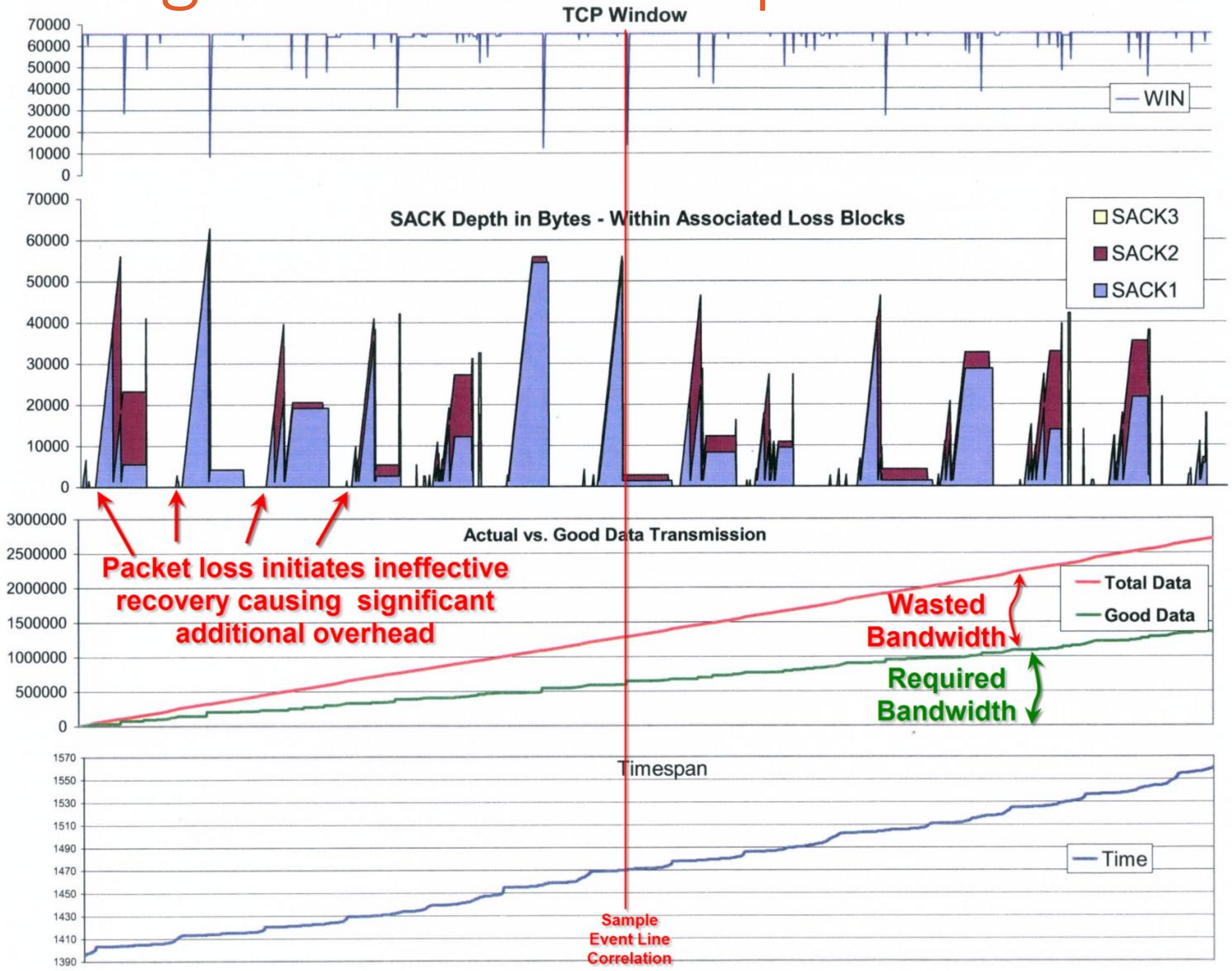
50 year old TCP's Idiosyncrasies

SOME REASONS FOR QUIC

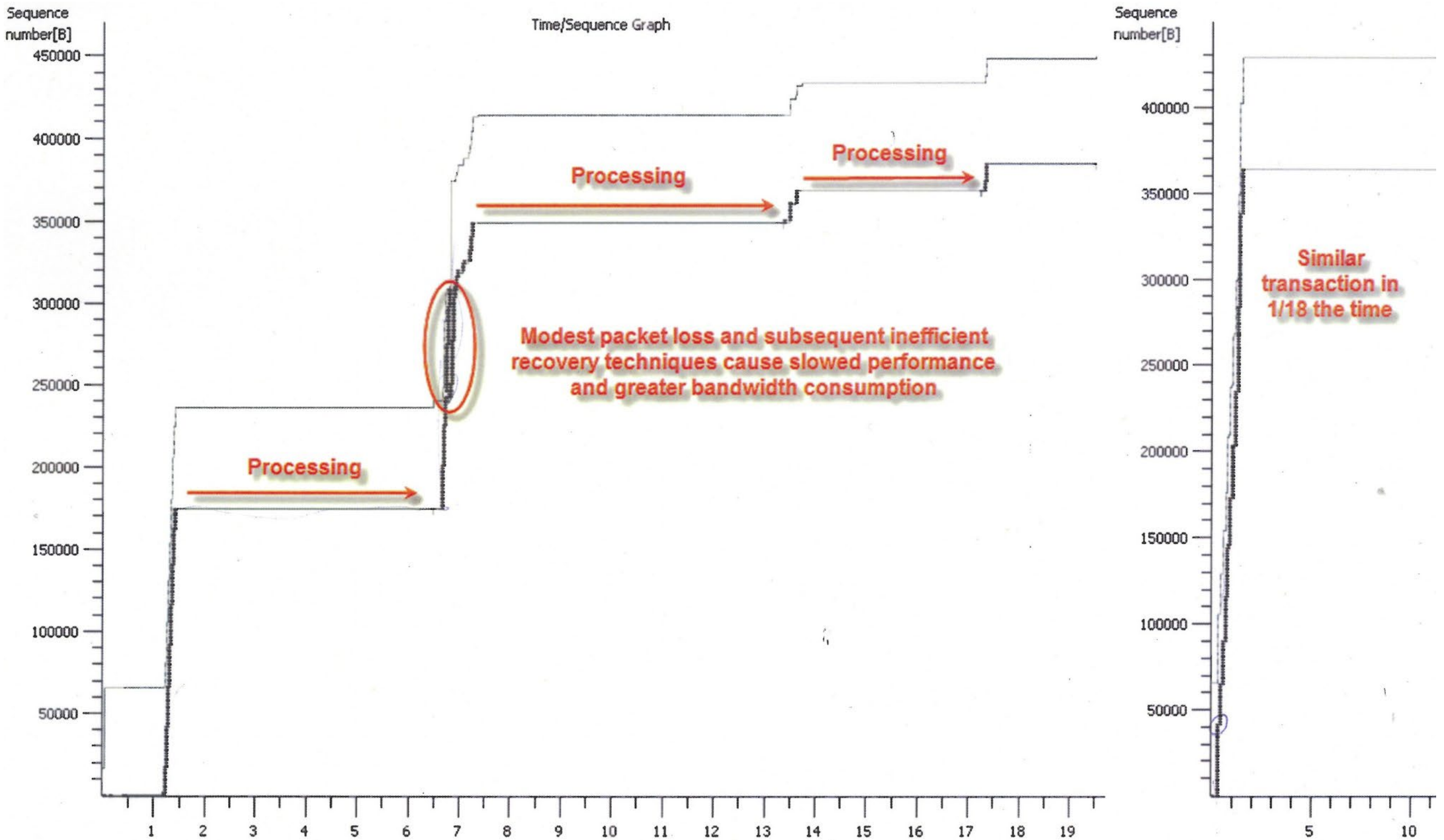
TCP Data Duplication Details



Significant Data Duplication

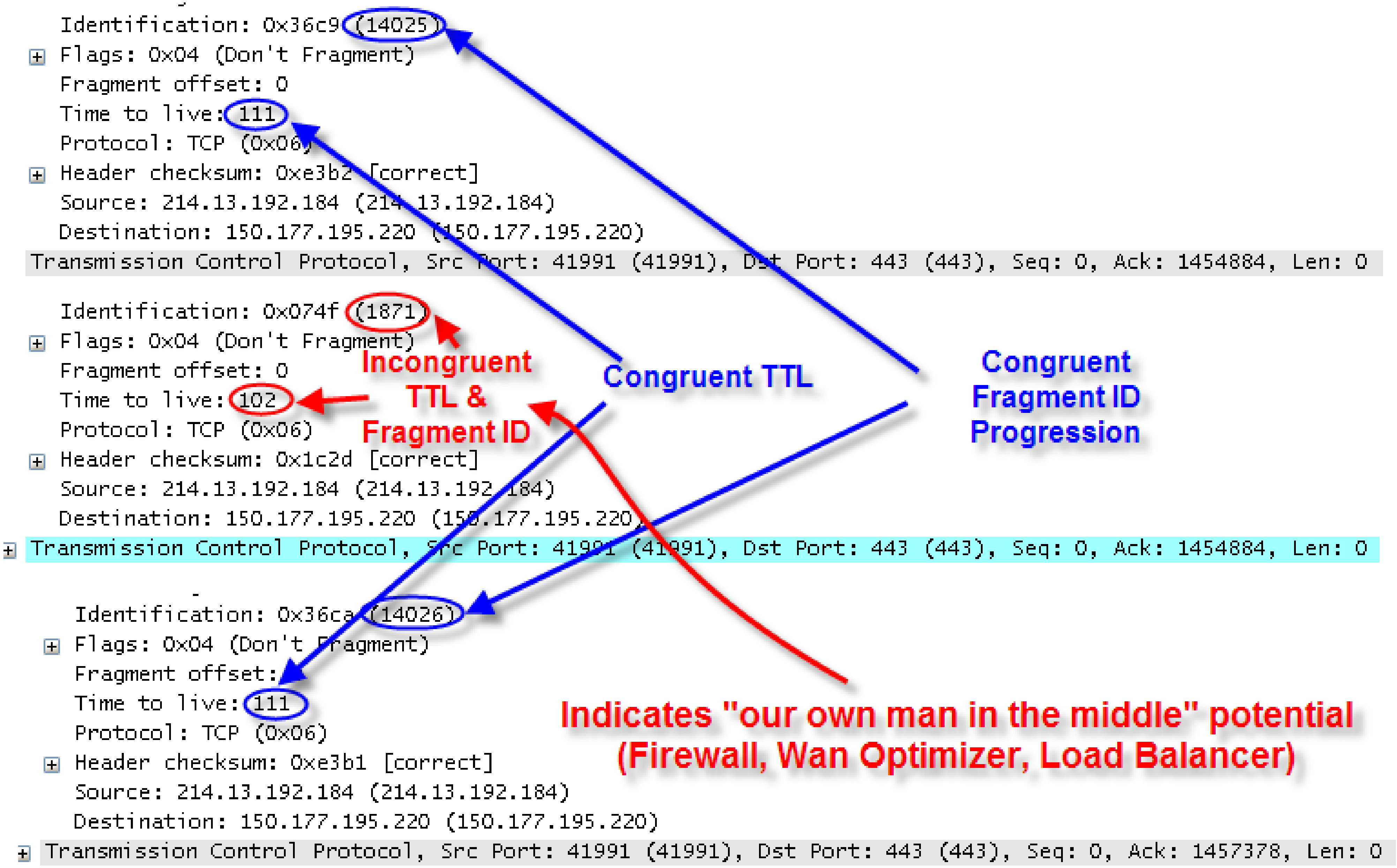


Data Duplication & App Processing

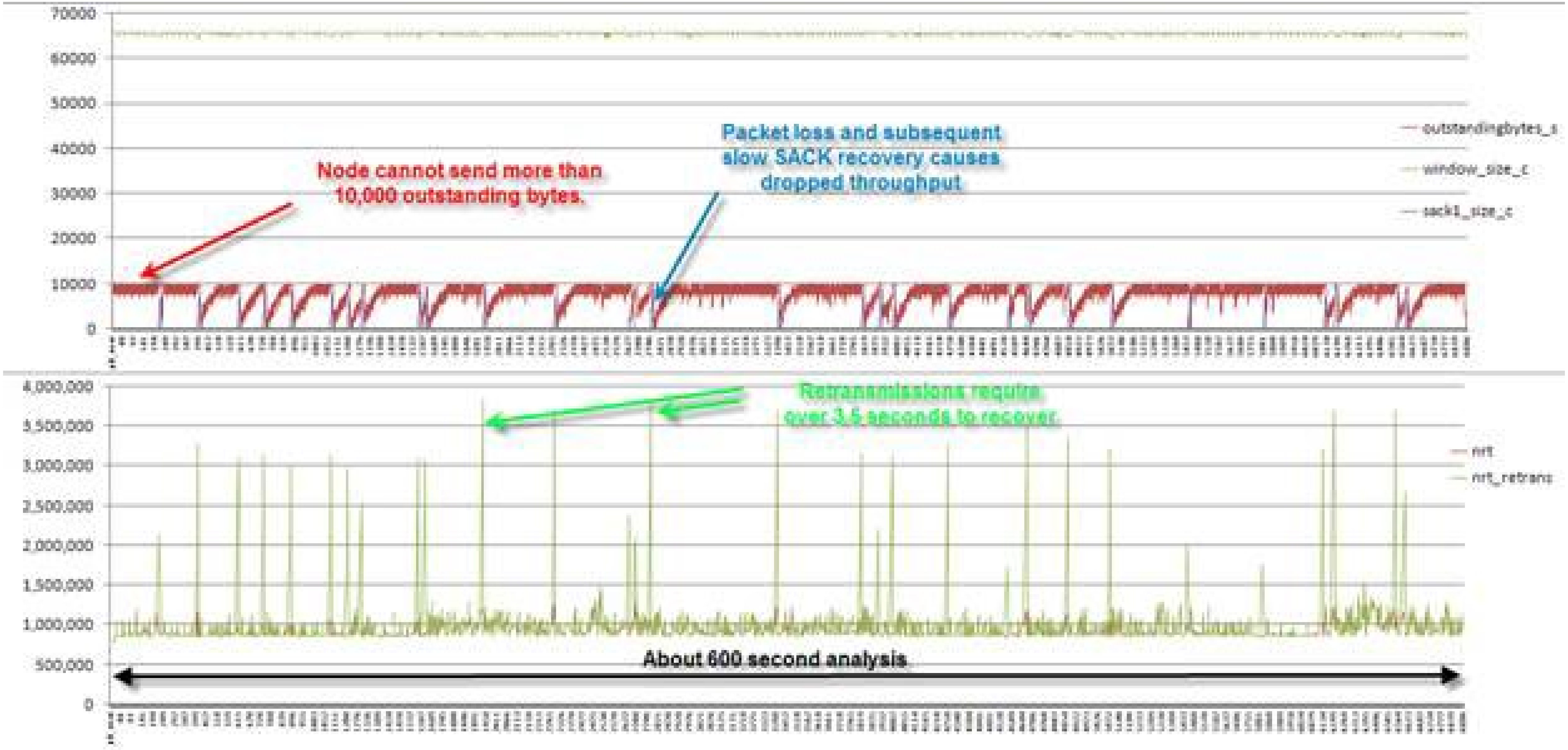


TCP – Packet Loss – Poor Recovery

HOP/TTL Incongruity “our own man in the middle”

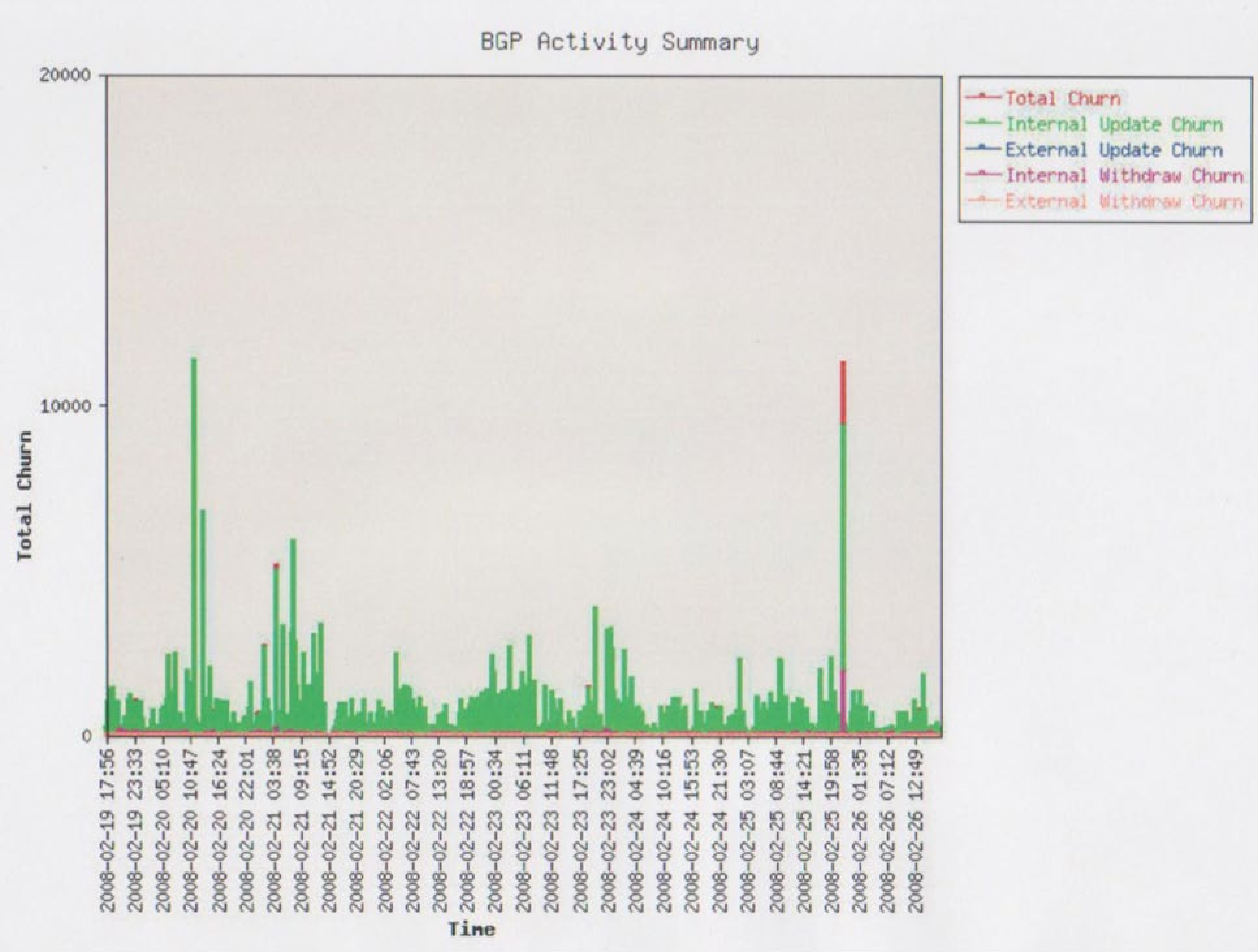


TCP – Session Performance

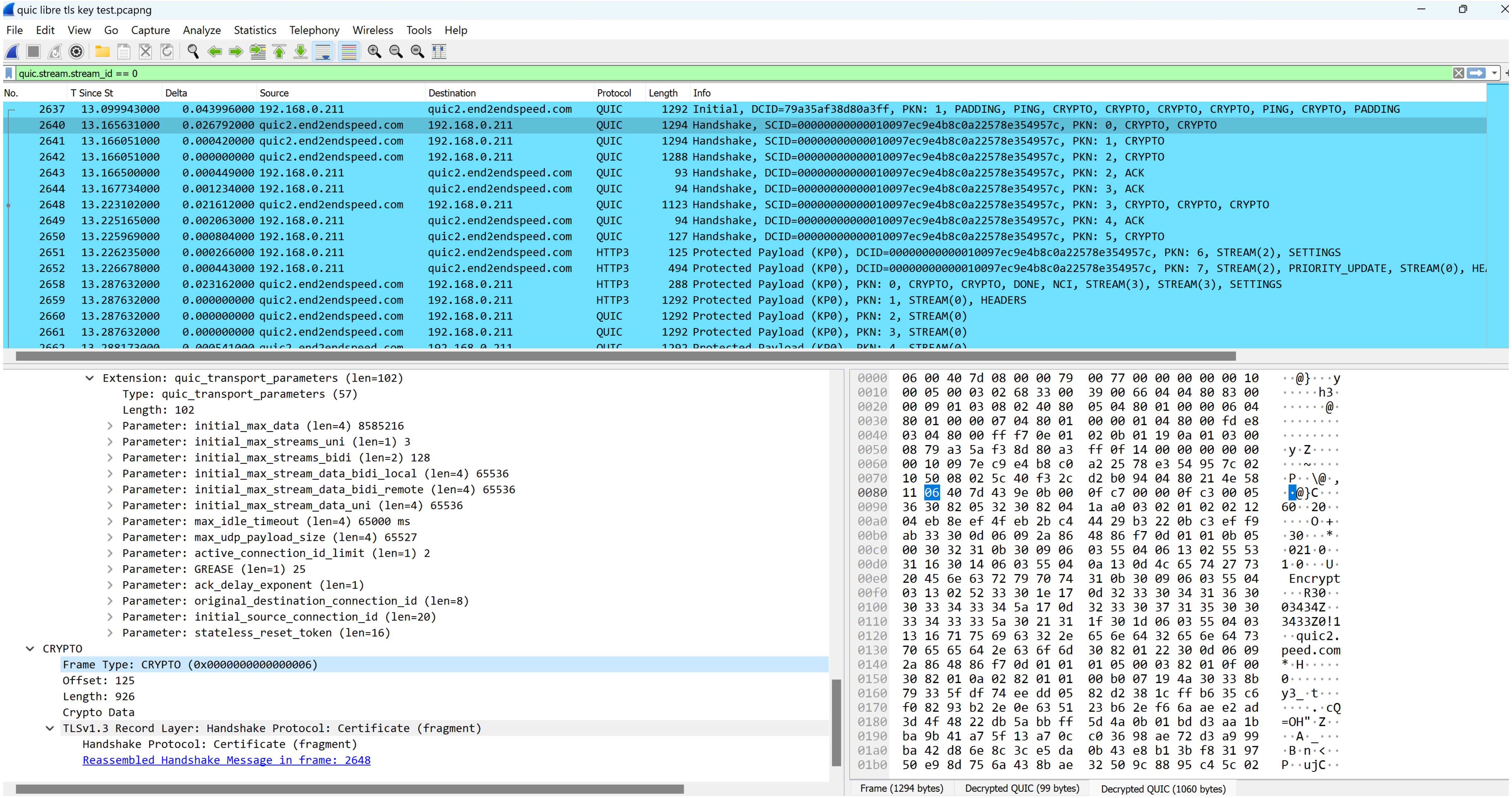


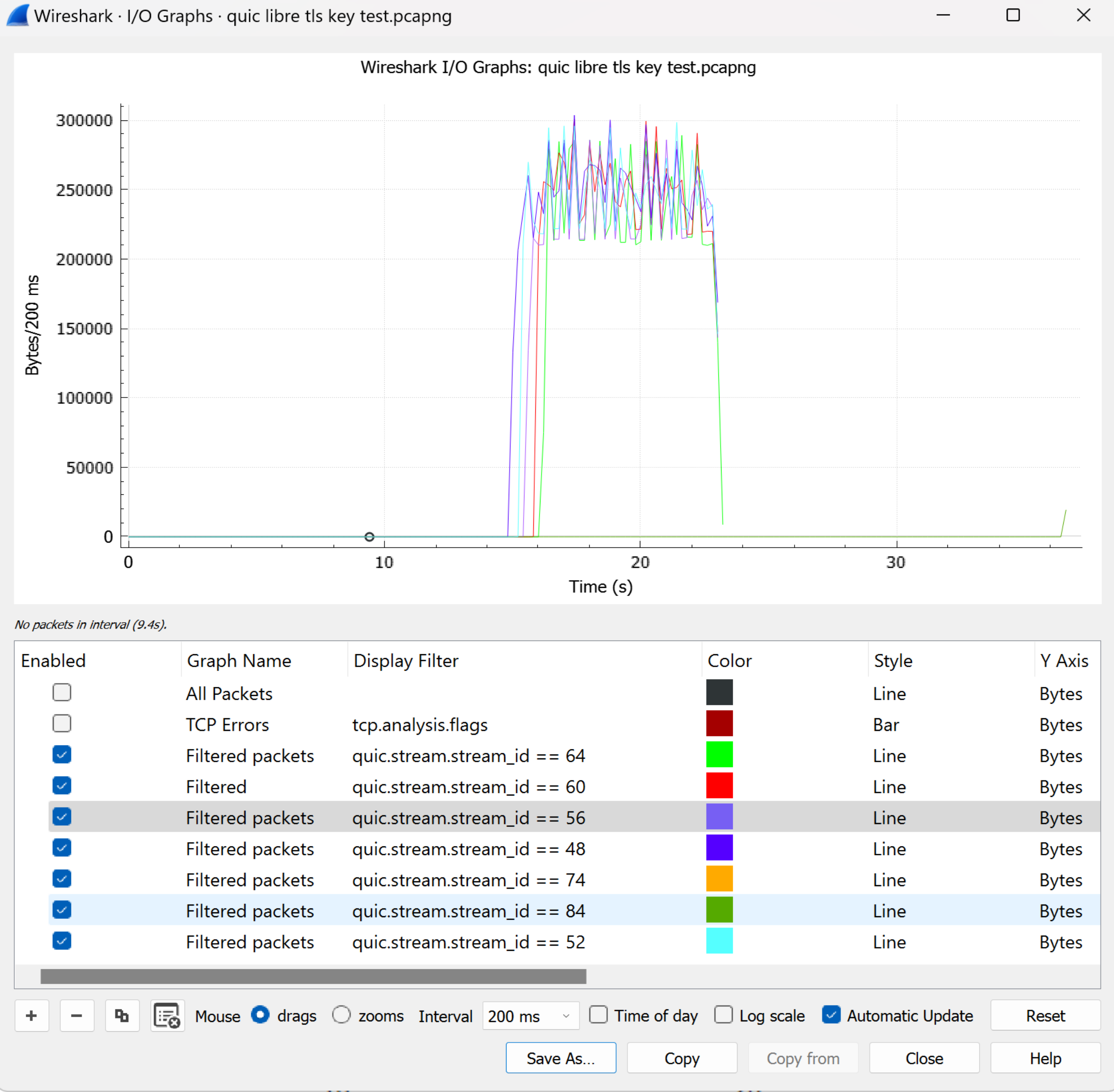
NAT, PAT or Route Changes Impact on Sessions

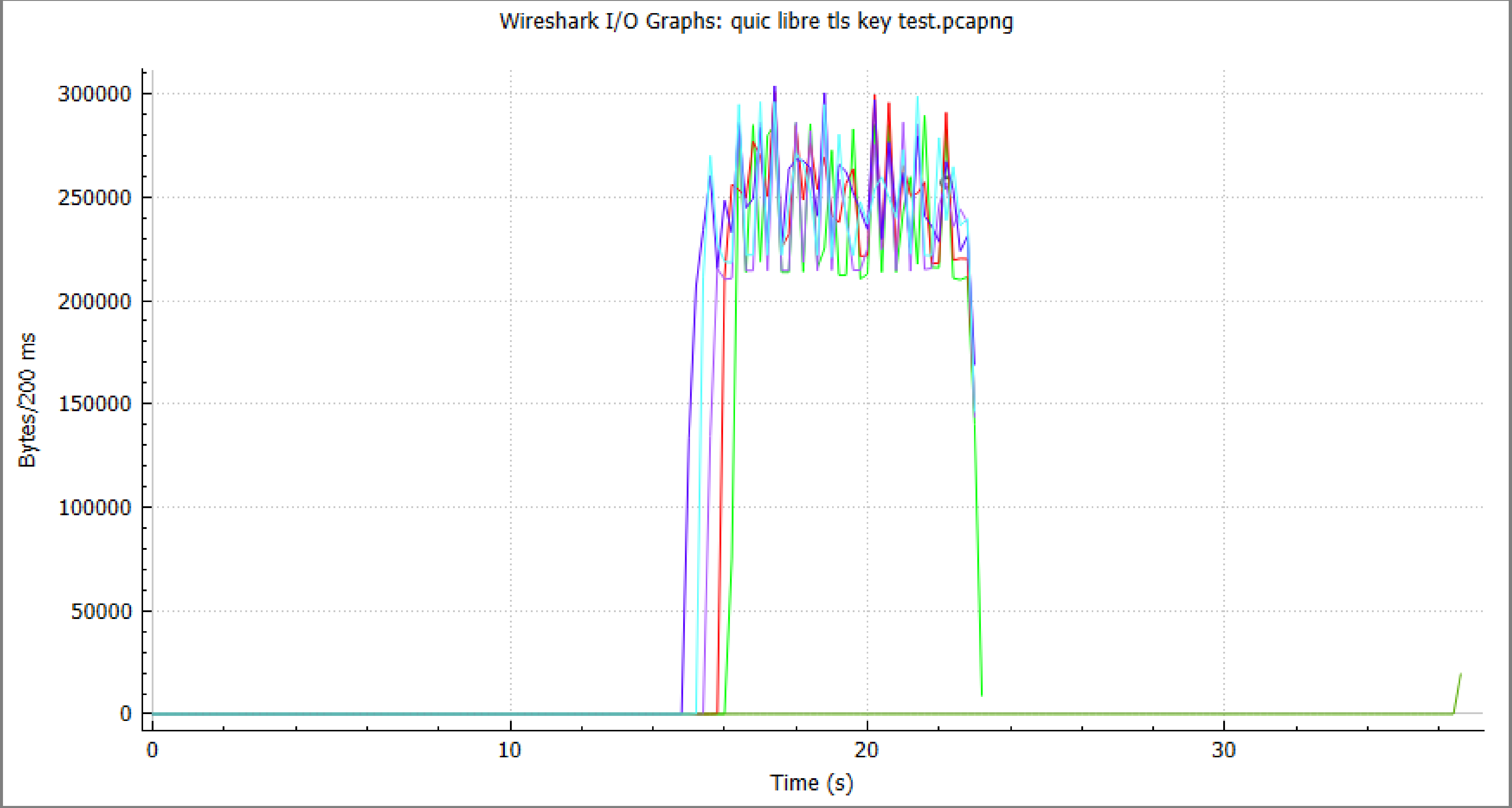
Instability of routing metrics



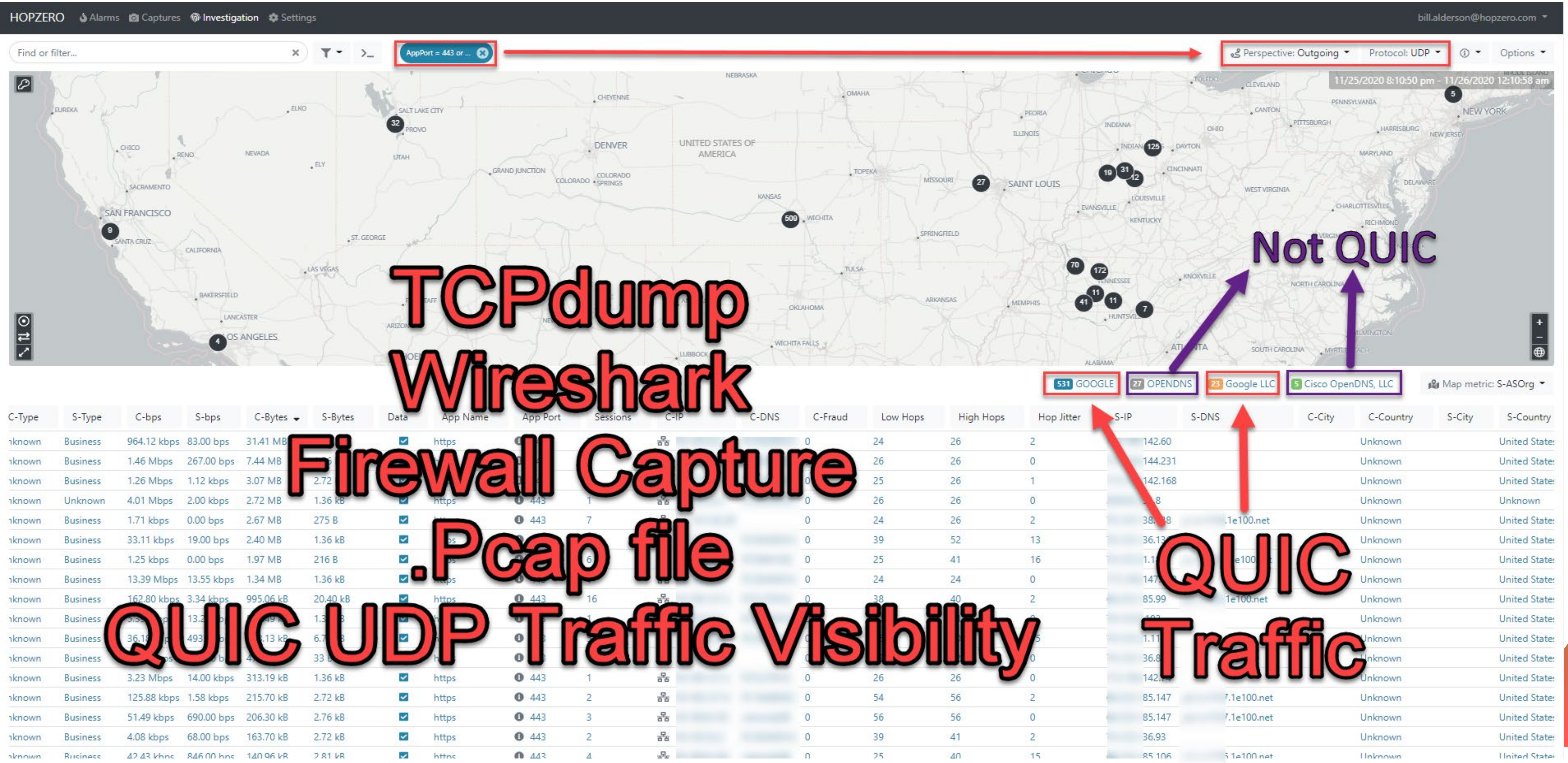
QUIC Decrypted




Wireshark · I/O Graphs · quic libre tls key test.pcapng
Wireshark I/O Graphs: quic libre tls key test.pcapng
Bytes/200 ms
300000
250000
200000
150000
100000
50000
0
0
10
20
30
Time (s)
No packets in interval (9.4s).
Enabled
Graph Name
Display Filter
Color
Style
Y Axis
All Packets
Line
Bytes
TCP Errors
tcp.analysis.flags
Bar
Bytes
Filtered packets
quic.stream.stream_id == 64
Line
Bytes
Filtered
quic.stream.stream_id == 60
Line
Bytes
Filtered packets
quic.stream.stream_id == 56
Line
Bytes
Filtered packets
quic.stream.stream_id == 48
Line
Bytes
Filtered packets
quic.stream.stream_id == 74
Line
Bytes
Filtered packets
quic.stream.stream_id == 84
Line
Bytes
Filtered packets
quic.stream.stream_id == 52
Line
Bytes
Mouse
drags
zooms
Interval
200 ms
Time of day
Log scale
Automatic Update
Reset
Save As...
Copy
Copy from
Close
Help


Wireshark I/O Graphs: quic libre tls key test.pcapng
Bytes/200 ms
300000
250000
200000
150000
100000
50000
0
0
10
20
30
Time (s)

How to identify existing QUIC Users



UDP vs TCP vs QUIC Firewall

QUIC uses UDP Port 443 to Servers

UDP firewall state is basic compared to TCP

QUIC encrypts everything above UDP except small part of initial packets containing Link ID

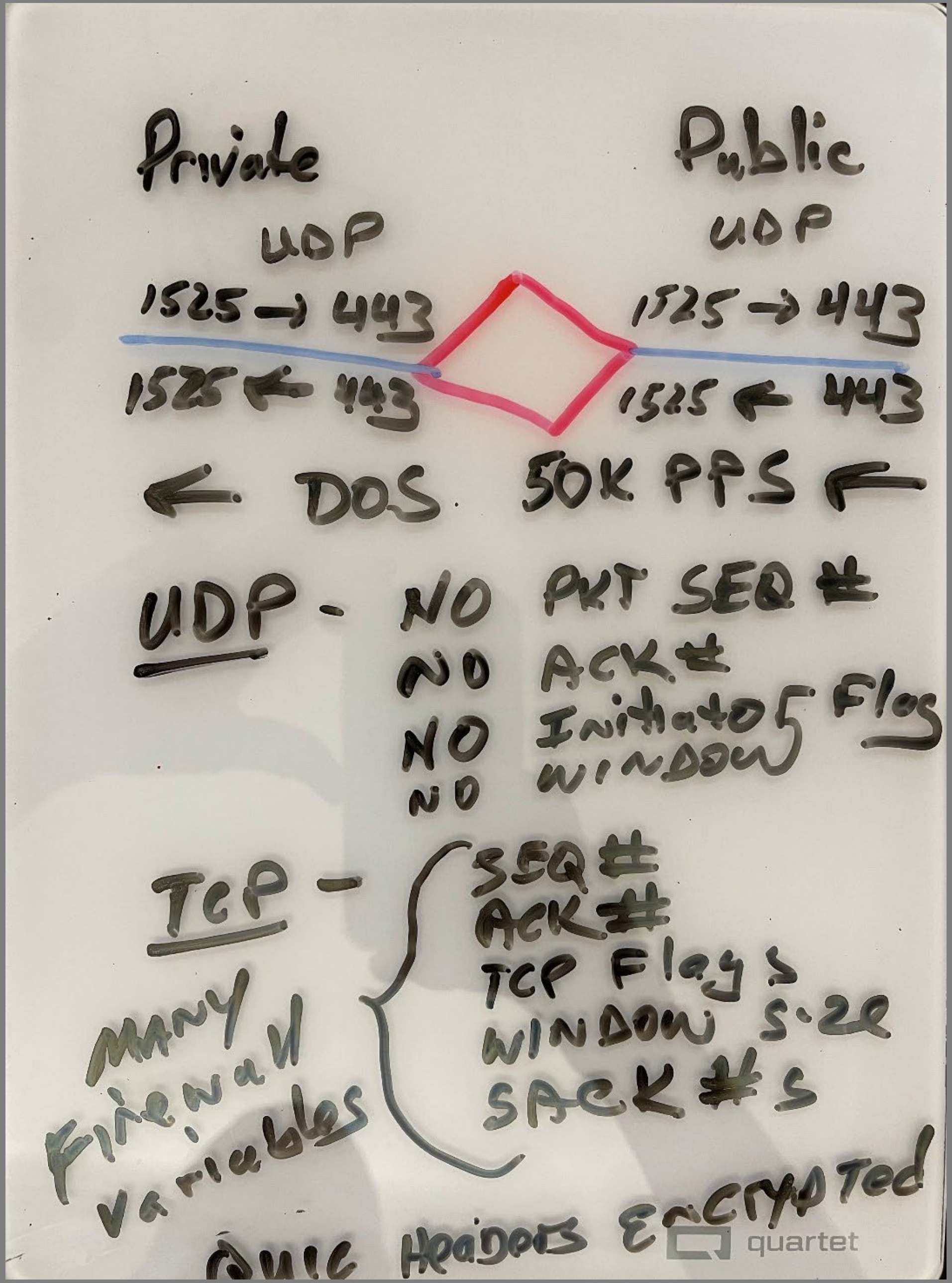
Firewalls blind - cannot inspect protocol or content!

Apps- Browsers now, but many more uses coming

QUIC overcomes NAT issues

It only cares about Link ID’s, not Port#’s not IP’s

Sends large MTU initial packet sizes



What to do next?

Personal / Home:
A.) Update Router-Firewall
B.) Use QUIC carefully
C.) Join SecurityInstitute.com QUIC Protocol Space

Work / School:
A.) Inform IT Security about QUIC working, or not.
B.) Careful they don't "shoot the messenger"

Security Pro's:
A.) Learn tools to identify QUIC
B.) Become Certified in QUIC https://SecurityInstitute.com

Vendors:
A.) Let us know about QUIC supported products
B.) Tell us about new QUIC products coming
C.) Sponsor and participate in SecurityInstitute.com QUIC

Triple+ Web Performance

at a Price...

3x FASTER

SECURITY THREATS

What Will You Do?

QUIC

Decrypting TLS & QUIC Headers

To record QUIC session information including encryption keys, you can use the SSLKEYLOGFILE environment variable. This method is supported by many TLS libraries, such as OpenSSL and BoringSSL, which are often used in QUIC implementations. The SSLKEYLOGFILE environment variable specifies a file path where the TLS session secrets will be written, enabling decryption of QUIC traffic for analysis and diagnostic purposes.

Here's how to use the SSLKEYLOGFILE method:

1.Set the SSLKEYLOGFILE environment variable to the desired file path before starting the client or server application that uses QUIC:

For Linux and macOS:

javascript

export SSLKEYLOGFILE=/path/to/your/sslkeylogfile.txt

For Windows:

vbnet

set SSLKEYLOGFILE=C:\path\to\your\sslkeylogfile.txt

2.Run the client or server application that uses QUIC. The application will write the TLS session keys to the specified file as they are established.

3.Use the collected keys to decrypt and analyze the QUIC traffic. Tools like Wireshark can use these keys to decode encrypted QUIC streams in captured packet data.

Remember that handling encryption keys securely is critical, as exposing these keys can compromise the security and privacy of the QUIC sessions. Only use this method for diagnostic purposes and with the appropriate permissions, and always follow best practices for handling sensitive data.

*Some Chat GPT helped find and organize some of this text.

QUIC Improvements over TCP

Head-of-line (HOL) blocking is a problem that occurs in TCP when a lost or delayed packet prevents the processing of subsequent packets in the same data stream, causing increased latency and reduced performance. QUIC solves the head-of-line blocking problem more effectively than TCP by employing the following techniques:

1. Independent streams: QUIC uses multiplexed streams over a single connection, where each stream is independent of the others. This means that if a packet is lost or delayed in one stream, it does not affect the other streams. In contrast, TCP treats all data within a connection as a single, ordered byte stream, which means that any packet loss or delay can block the entire connection.

2. Stream-level error correction: QUIC handles error correction and retransmissions at the stream level, rather than at the connection level, as in TCP. This enables QUIC to recover from packet losses in one stream without affecting other streams, further reducing the impact of head-of-line blocking.

3. Faster packet retransmissions: QUIC can retransmit lost packets more quickly than TCP because it does not rely on a single, global retransmission timer. Instead, QUIC uses per-packet timers and can quickly detect and retransmit lost packets without waiting for a full round-trip time, as is typically the case with TCP.

4. Selective acknowledgments: QUIC uses selective acknowledgments (ACKs) to inform the sender about received packets, as well as any gaps in the sequence of received packets. This allows the sender to quickly identify lost packets and retransmit them, reducing the impact of head-of-line blocking.

By using these techniques, QUIC effectively mitigates the head-of-line blocking problem, leading to better performance, reduced latency, and improved user experience, especially in environments with high packet loss or network congestion.

0-RTT Data Request Response Size

In a 0-RTT (Zero Round-Trip Time) session, the amount of data that can be sent in the initial request depends on the server's maximum allowed 0-RTT data size, which can vary depending on the server's configuration and preferences. There isn't a fixed theoretical capacity for all cases, as it depends on the server's specific settings.

However, it's important to note that 0-RTT data should generally be limited to a small amount, as sending large amounts of data in the initial request could increase the risk of replay attacks. The server must enforce proper anti-replay measures and limit the use of 0-RTT data to mitigate this risk.

In practice, 0-RTT data is typically used for non-sensitive, idempotent requests like HTTP GET requests or other actions that can be safely retried without causing unintended side effects. This ensures that even if a replay attack occurs, the consequences are minimal.

In TLS 1.3, the "max_early_data_size" parameter within the "NewSessionTicket" message specifies the maximum amount of 0-RTT data a client can send during a 0-RTT session. The "max_early_data_size" is a 32-bit unsigned integer, so the maximum value that can be represented is 2^32 - 1 bytes, which is equal to 4,294,967,295 bytes or approximately 4 GiB.

However, it's important to remember that setting such a high limit for 0-RTT data is not recommended, as it could increase the risk of replay attacks. In practice, servers are likely to set much smaller limits to ensure security and protect against potential abuse.

QUIC Frame Concept

The QUIC protocol uses a modular and extensible framing mechanism, which allows for the efficient encoding of different types of data while also providing flexibility for future enhancements. Some common types of frames in QUIC include:

STREAM frames: These frames carry application data between endpoints and are used for reliable, in-order transmission of data within a specific QUIC stream.

ACK frames: These frames are sent by the receiving endpoint to acknowledge the receipt of one or more packets, indicating the packets' sequence numbers and any gaps (i.e., lost or delayed packets).

MAX_DATA and MAX_STREAM_DATA frames: These frames are used for flow control, with MAX_DATA controlling the overall amount of data that can be sent across all streams and MAX_STREAM_DATA controlling the amount of data that can be sent within a specific stream.

RESET_STREAM frames: These frames are sent by an endpoint to indicate that it wants to abruptly terminate a stream without completing the transmission of all data.

CONNECTION_CLOSE and APPLICATION_CLOSE frames: These frames are used to signal the termination of a QUIC connection, either due to an error or a graceful shutdown initiated by the application.

PING frames: These frames are used to test the connection's liveness and to keep the connection alive in the presence of idle timeouts.

By using frames to carry various types of information, QUIC enables efficient, flexible, and extensible communication between endpoints while maintaining performance and security.

Introduction to QUIC for Network and Security Technologists

QUIC (Quick UDP Internet Connections) is a transport layer protocol started by Jim Roskind at Google (Now AWS) to improve the security, performance, and reliability of web connections. QUIC uses UDP as its transport protocol, providing faster connection establishment, reduced latency, and built-in encryption.

Internet Engineering Task Force IETF changed its name to QUIC – no acronym to lose its Google roots. Greatly enhancing and integrating with TCP features.

Encryption and security: QUIC incorporates Transport Layer Security (TLS) 1.3, ensuring all transmitted data is encrypted by default. This enhances security compared to older HTTP/2 connections, which do not always require encryption.

Faster connection establishment: QUIC reduces the number of round trips required to establish a secure connection, resulting in a faster and more efficient process compared to traditional TCP/TLS connections.

0-RTT connection resumption: QUIC supports 0-RTT (Zero Round-Trip Time) connection resumption, allowing for faster reconnections between clients and servers that have previously communicated. This feature should be implemented with caution, as it can pose a risk of replay attacks.

Connection migration: QUIC allows for connection migration, enabling a client to change its IP address without losing the connection. This feature improves the stability of secure connections in mobile or unstable network environments.

Multiplexed streams and head-of-line blocking: QUIC's support for multiplexed streams can help mitigate head-of-line blocking, enhancing the performance and security of connections by reducing latency.

Forward error correction: QUIC uses forward error correction (FEC) to reduce the impact of packet loss, enhancing the reliability and security of connections.

Potential vulnerabilities: While QUIC is designed with security in mind, potential vulnerabilities exist, such as 0-RTT vulnerabilities, key update attacks, DoS attacks, Connection ID privacy concerns, and implementation flaws. Awareness and mitigation strategies are essential for ensuring optimal security.

Limited adoption and compatibility: QUIC is becoming widely adopted, with more implementations monthly. Network and security technologists should be prepared to work with both QUIC-enabled and non-QUIC environments.

QUIC’s Top 5 Security Vulnerabilities

0-RTT vulnerabilities: The 0-RTT (Zero Round-Trip Time) feature can make QUIC connections more susceptible to replay attacks. An attacker may intercept and replay a 0-RTT connection attempt to gain unauthorized access. To mitigate this risk, servers should enforce proper anti-replay measures and limit the use of 0-RTT data.

Key update attacks: QUIC's key update mechanism, which periodically updates encryption keys, could be exploited by attackers to force clients or servers to use weak or compromised keys. This issue can be addressed by implementing proper key management practices and ensuring that keys are securely generated and stored.

Denial of Service (DoS) attacks: QUIC's reliance on the User Datagram Protocol (UDP) could make it more susceptible to DoS attacks. Attackers might flood a server with malformed or large packets to exhaust its resources. Server operators should employ rate limiting, filtering, and other techniques to prevent such attacks.

Connection ID privacy concerns: QUIC's use of Connection IDs to maintain sessions can improve privacy but may also be exploited by attackers to track users across different connections. Ensuring that Connection IDs are generated and managed securely can help minimize this risk.

Implementation flaws: As with any protocol, security issues may arise due to flaws in the implementation of QUIC by software developers. To address this, it is essential to use well-tested and regularly updated libraries, adhere to best practices, and perform thorough security audits and testing of QUIC-enabled applications.

QUIC: Top 10 Things to Know

Encryption by default: QUIC incorporates built-in encryption using Transport Layer Security (TLS) 1.3, ensuring that all data transmitted is secure by default. This is an improvement over HTTP/2, which does not require encryption.

Connection establishment: QUIC reduces the number of round trips required to establish a secure connection, speeding up the process and making it more efficient.

0-RTT connection resumption: QUIC allows for 0-RTT (Zero Round-Trip Time) connection resumption, enabling faster reconnections between clients and servers that have previously communicated. This can, however, pose a risk of replay attacks if not properly implemented.

Improved privacy: QUIC's connection identifiers do not reveal user IP addresses, making it harder for eavesdroppers to track users across different connections and improving privacy.

Resistance to replay attacks: QUIC has built-in mechanisms to counter replay attacks, but proper implementation is essential to ensure the security of the protocol.

Connection migration: QUIC supports connection migration, allowing a client to change its IP address without losing the connection. This can help maintain secure connections, even in mobile or unstable network environments.

Forward error correction: QUIC uses forward error correction (FEC) to reduce the impact of packet loss, enhancing reliability and security.

Reduced impact of head-of-line blocking: QUIC's multiplexed streams can help mitigate head-of-line blocking, improving the performance and security of connections.

Key QUIC vs. TCP Improvements

Head-of-line (HOL) blocking is a problem that occurs in TCP when a lost or delayed packet prevents the processing of subsequent packets in the same data stream, causing increased latency and reduced performance. QUIC solves the head-of-line blocking problem more effectively than TCP by employing the following techniques:

1. **Independent streams:** QUIC uses multiplexed streams over a single connection, where each stream is independent of the others. This means that if a packet is lost or delayed in one stream, it does not affect the other streams. In contrast, TCP treats all data within a connection as a single, ordered byte stream, which means that any packet loss or delay can block the entire connection.
2. **Stream-level error correction:** QUIC handles error correction and retransmissions at the stream level, rather than at the connection level, as in TCP. This enables QUIC to recover from packet losses in one stream without affecting other streams, further reducing the impact of head-of-line blocking.
3. **Faster packet retransmissions:** QUIC can retransmit lost packets more quickly than TCP because it does not rely on a single, global retransmission timer. Instead, QUIC uses per-packet timers and can quickly detect and retransmit lost packets without waiting for a full round-trip time, as is typically the case with TCP.
4. **Selective acknowledgments:** QUIC uses selective acknowledgments (ACKs) to inform the sender about received packets, as well as any gaps in the sequence of received packets. This allows the sender to quickly identify lost packets and retransmit them, reducing the impact of head-of-line blocking.

QUIC Encryption Explained vs TCP

QUIC packet header encryption is a mechanism that protects certain parts of the QUIC packet header from being observed or modfied by third parties, such as middleboxes or eavesdroppers. This enhances privacy and security compared to traditional transport protocols like TCP, where some header information remains exposed.

In QUIC, the packet payload and certain parts of the header are encrypted together using the same encryption keys. The payload is encrypted using modern cryptographic algorithms like AES-GCM or ChaCha20-Poly1305, which also provide authentication.

Not all parts of the QUIC header are encrypted. The packet number, for example, remains in the clear. The reason is to allow for better handling of packet loss and reordering, as the packet number helps identify which packets have been received and which ones are still missing. QUIC:

1. **Encrypts the payload:** The payload data (e.g., application data) is encrypted using a symmetric key negotiated during the QUIC handshake.

2. **Protect specific header fields:** QUIC protects certain header fields, such as the Key Phase, Spin Bit, and some reserved bits, using a technique called "header protection." This is done by generating a header protection mask based on the packet encryption key and the unprotected header.

3. **Apply the header protection mask:** The header protection mask is XORed with the specific header fields that need to be protected. This process encrypts these fields and prevents them from being observed or modified by third parties.

4. **QUIC uses TLS for header encryption:** The header protection mechanism is built into the QUIC protocol itself. As a result, there's no "second encryption" layer for the header compared to the payload. The encryption keys for both payload and header protection are derived from the same initial secret negotiated during the QUIC handshake.

5. **QUIC uses per packet encryption vs. TCP Stream-based encryption:** When TCP is combined with TLS, it provides stream-based encryption, which means that the entire data stream is encrypted as a whole rather than on a per-packet basis. This can make it more challenging to handle packet loss or reordering, as lost or out-of-order packets can cause the entire stream to stall until the missing packet is received.

6. **TCP has Exposed headers:** In, some header information remains exposed, which can potentially be exploited by attackers or used for network analysis by third parties. This can be a privacy and security concern compared to QUIC's header protection.

7. **TCP does not natively support connection migration:** If a user changes their network connection (e.g., switching from Wi-Fi to cellular data), the existing TCP connection/s must be terminated, and a new connection needs to be established, causing additional latency and potential disruptions.

QUIC vs. TCP Encryption

Per-packet encryption: QUIC encrypts every packet individually with packet numbers in the clear. This allows for better handling of packet loss and reordering compared to TCP. QUIC uses modern cryptographic algorithms such as AES-GCM or ChaCha20-Poly1305 for encryption and authentication.

Packet header protection: QUIC also protects packet headers from being observed or modified by third parties. This enhances privacy and security while preventing potential attacks that could exploit exposed header information.

Connection migration: QUIC supports connection migration, which means that a connection can be transferred between IP addresses without breaking the connection. This can be useful in cases of network changes or mobility (e.g., when a user switches from Wi-Fi to cellular data). Per-packet encryption enables this feature, as packets can be independently decrypted and processed.

TCP encryption (with TLS):

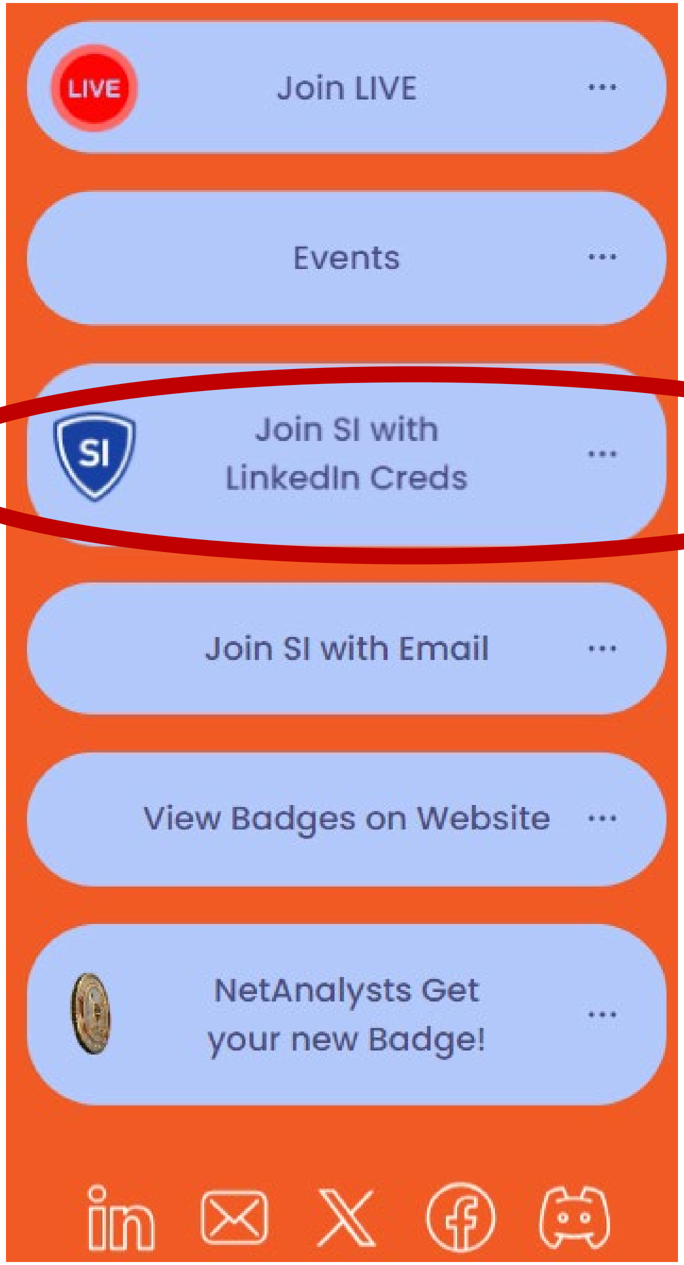
Protocol: TCP is built on top of IP and provides a reliable, ordered, and error-checked delivery of data between applications. TCP is the foundation for many application-level protocols, including HTTP, HTTPS, and FTP.

Stream-based encryption: When TCP is combined with TLS, it provides stream-based encryption, which means that the entire data stream is encrypted as a whole rather than on a per-packet basis. This can make it more challenging to handle packet loss or reordering, as lost or out-of-order packets can cause the entire stream to stall until the missing packet is received.

Exposed headers: In TCP, some header information remains exposed, which can potentially be exploited by attackers or used for network analysis by third parties. This can be a privacy and security concern compared to QUIC's header protection.

Connection migration limitations: TCP does not natively support connection migration. If a user changes their network connection (e.g., switching from Wi-Fi to cellular data), the existing TCP connection must be terminated, and a new connection needs to be established, causing additional latency and potential disruptions.

SECURITY INSTITUTE



Topics Prof Assn's Conferences SME's Vendors
Content Videos LiveStream Collaboration
Root Cause Analysis Chat GPT Cybersecurity
QUIC Protocol SharkFest - WireShark Betty Dubois
ISSA / ISC2 Leadership Podcasts

Packetman007