**QUIC vs TCP – Summary**

# Comparison of QUIC vs TCP

|  |  |  |
| --- | --- | --- |
| Feature/Issue | TCP | QUIC |
| Base Protocol | TCP (Transmission Control Protocol) | UDP (User Datagram Protocol) |
| Connection Setup | 3-way handshake | Combined handshake and TLS setup in 1-RTT message.  **0RTT** – if the client previously communicated with the server, it can use the connection ID to immediately send data |
| Packet Numbering | The numbering is related to the data size by bytes. The number field is of size 4 bytes | The numbering is monotonically.  The number field is of size 8 bytes |
| Connection Persistence | IP address sensitive; requires a new handshake on IP change | Uses Connection ID to maintain the connection even if the IP changes |
| Packet Loss Handling | **HOL – Head Of Line blockage:**  Lost packets block subsequent packets until retransmitted | Handles packet loss by only blocking streams with lost packets until retransmission |
| Protocol Header | Fixed-size header; limited sequence and ACK fields, small flow control window. Can cause resetting of packet’s number in higher speed network | Larger packet size for ACK frame; flexible packet format with long and short headers for an efficient data transmission |
| Stream Multiplexing | Not inherently supported | Supports multiple simultaneous streams within a single connection; each stream identified by a stream ID |
| Packet Loss Detection | Based on unacknowledged packets | Based on packet number or stream frame offset, with thresholds for declaring loss |
| Loss Recovery | Retransmits lost packets, impacting all subsequent packets | Retransmits lost packets over their original streams, impacting only the certain streams. |
| Congestion Control | Relies on packet loss to signal congestion.  Supports algorithms for CC management | Uses stream ID, frame offset, packet number and sample RTT to check thresholds to detect packet loss / persistent congestion.  Supports algorithms for CC management |
| NewReno Algorithm | - Slow Start: Exponential window growth until loss  - Recovery: Window halved, then gradual increase  - Congestion Avoidance: Slow window growth until loss  - Persistent Congestion: Resets to slow start | My group did not implement Congestion Control management. |
| Real-Time Data Delivery | Not optimized for real-time applications | Supports unreliable but secure data delivery with DATAGRAM frames; acknowledgments track packet loss but does not retransmit them. |

**TCP problems:**

* Mismatch between unacknowledged packets and network congestion - The number of unacknowledged packets (those that have been sent but not yet acknowledged) does not accurately reflect the number of packets currently inside the network.
* Constrained window advancement on packet loss – in TCP, a lost packet will block the line for the other packets until retransmitted.
* Long connection set up – a 3-way handshake
* Fixed protocol header – too small size for sequence and ACK fields (They can reach the limit and then reset at high network speed). Flow control is up to 65KB, thus capping the throughput.
* IP address sensitive – once the IP address changes, the data which was exchanged so far is lost, and a new 3-way handshake is required to set up the new connection.

**QUIC:**

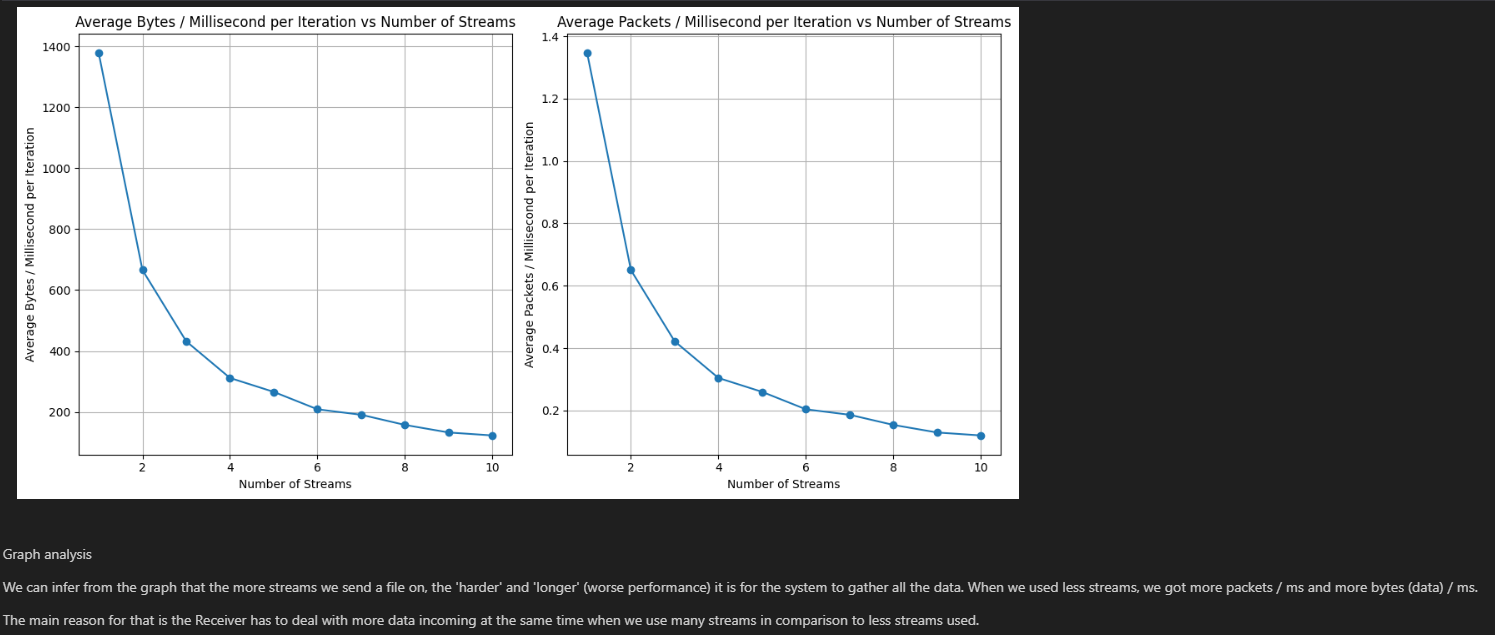
* UDP based connection
* Connection ID – using an ID to define each address so that if IP changes, the packets will still be delivered to the right recipient. After a network change, a migrating end point can send a packet with the established ID. The other endpoint will send a message, waiting for the same message to be sent back to it for validation.
* Handshake – QUIC combines the information gathering and TLS (encrypting) parts into 1-RTT message. Reduces the time for connection set up.
* Packet – the long format is used during connection set up, then the short format during data transmission. Each packet is assigned a unique packet number that increases monotonically to easily deal with packet loss. That way QUIC does not have the mismatch issue from TCP. The packet also has bigger size for ACK frame to prevent resetting.

**Special QUIC features:**

* Stream – each QUIC connection can have multiple flows simultaneously. Each stream is identified with stream ID. Stream has frame offset to detect packet loss. Dealing with Head Of Line blockage due to packet loss by making only streams with data frames contained in the lost packet wait for retransmission.
* Datagrams – a QUIC can support datagram, a type of packet which will not be retransmitted upon packet loss. However, this packet will still get the ACK to track the packet loss.
* Congestion Control – QUIC identifies congestion by packet number , and packet loss by stream frame offset. Quic does not collapse the congestion window unless detected persistent congestion. Quic uses its RTT samples to measure the expected time for an ACK, on which it relies to identify packet loss.
* Packet loss – A packet is considered lost when a later-sent packet is already acknowledged, or when a threshold is met.
* Packet loss threshold – either by packet number or by time. Packet number; meaning by a given limit t, if we acknowledge t packets that were sent after the supposed ‘lost packet’, and they were all acknowledged, then the packet is lost. By time; by comparing the time since it was sent to the sample RTT.
* Loss recovery – the lost frames will be put in a new packet, with a new packet number unrelated to lost packets.

**The NewReno algorithm:**

* Slow start – this state is entered when a persistent congestion occurs. During this state, the congestion window will grow explonentially until a packet loss is declared. Which will then start the Recovery state.
* Recovery – reduces the congestion window by half, and sets it as the new threshold. Then, once a packet that is sent has been acknowledged, the Congestion Avoidance state will start.
* Congestion Avoidance – Increases the window size (slower than the start state, and slowing down each time) until a packet loss is declared, which will then start the Recovery state.
* Persistent Congestion – if a persistent congestion occurs, which means we keep losing packets, then we restart the process – back to slow start.

**Streams conclusion:**

**Graph analysis**

We can infer from the graph that the more streams we send a file on, the 'harder' and 'longer' (worse performance) it is for the system to gather all the data.

When we used less streams, we got more packets / ms and more bytes (data) / ms.

The main reason for that is the Receiver has to deal with more data incoming at the same time when we use many streams in comparison to less streams used.