Slide 1: Introduction to QUIC

Title: *The Future of Transport Protocols: QUIC*
Content:

- QUIC (Quick UDP Internet Connections) is a modern transport layer protocol.
- Developed by Google and standardized by the IETF as RFC 9000.
- Combines the best features of TCP, UDP, TLS, and HTTP/2.
- Backbone of HTTP/3 and future web communication.

Slide 2: Key Features of QUIC

Title: *Features That Make QUIC Stand Out*

1. **Low Latency**:
    - Faster connection establishment using 0-RTT and combined handshake.
2. **Secure by Default**:
    - Built-in TLS 1.3 encryption ensures confidentiality and integrity.
3. **Multiplexing Without Blocking**:
    - Multiple streams per connection, avoiding head-of-line blocking.
4. **Connection Resilience**:
    - Seamless migration between networks (e.g., Wi-Fi to 4G).
5. **Built for Modern Internet**:
    - Optimized for mobile, video streaming, and gaming.

Slide 3: QUIC vs Traditional TCP

Title: *How QUIC Outperforms TCP*

Aspect	TCP	QUIC
Connection Setup	2-3 Round Trips (TCP+TLS)	1 Round Trip (or 0-RTT)
Encryption	Optional (TLS on top of TCP)	Always Encrypted (TLS 1.3)
Multiplexing	Head-of-Line Blocking	Independent Streams
Network Migration	Not Supported	Fully Supported
Performance	Good	Superior for modern apps

Slide 4: How QUIC Achieves These Benefits

Title: *Innovative Design Principles of QUIC*

- **UDP-Based**:
  - Uses lightweight UDP for faster transport and avoids TCP's legacy issues.
- **TLS Integration**:
  - Handshake merged with transport layer for reduced latency.
- **Stream Multiplexing**:
  - Multiple independent streams handled in a single connection.
- **Congestion Control**:
  - Advanced congestion and flow control mechanisms for smoother performance.
- **Packet Recovery**:
  - Loss recovery is faster, unlike TCP's retransmission delays.

---

1. **Faster Connection Setup**:
   - **TCP**: Requires a 3-way handshake (SYN-SYN/ACK-ACK) and an additional handshake for TLS encryption (2–3 round trips).
   - **QUIC**: Combines connection and encryption handshakes into a single step, achieving **1-RTT** (round-trip time) or even **0-RTT** for resumed connections.
   - **Benefit for Fixed Income**: Faster market data delivery and quicker order execution, critical in trading environments where milliseconds matter.
2. **No Head-of-Line Blocking**:
   - **TCP**: If a single packet is lost, all subsequent packets must wait until the lost packet is retransmitted, causing delays.
   - **QUIC**: Multiplexes streams independently within a connection, so a delay in one stream (e.g., bond price data) doesn't block others (e.g., yield updates).
   - **Benefit for Fixed Income**: Real-time updates of multiple data streams (prices, curves, spreads) without interruptions.
3. **Seamless Network Migration**:
   - **TCP**: Connections are tied to an IP address. If the network changes (e.g., Wi-Fi to 4G), the connection is dropped, and a new one must be established.
   - **QUIC**: Connections are tied to unique identifiers, not IP addresses. It allows seamless migration across networks.
   - **Benefit for Fixed Income**: Traders using mobile apps or switching networks during volatile markets can maintain uninterrupted connectivity.

4. **Always Encrypted**:
    - **TCP**: Requires optional TLS to encrypt connections, adding complexity and setup time.
    - **QUIC**: Encryption (TLS 1.3) is built into the protocol, ensuring that all communications are secure by default.
    - **Benefit for Fixed Income**: Protects sensitive financial data such as trading instructions, bond valuations, and client details without additional overhead.
5. **Reduced Latency**:
    - **TCP**: Designed decades ago for reliable delivery but lacks optimization for modern high-speed, low-latency needs.
    - **QUIC**: Optimized for high-speed data transfers with advanced congestion control and loss recovery mechanisms.
    - **Benefit for Fixed Income**: Reduces delays in trade execution and analytics API responses, enabling faster decision-making in high-frequency trading scenarios.

Internal Architecture of QUIC

1. Foundation: Built on UDP

- **Why UDP?**
    - UDP is a lightweight, connectionless protocol that doesn't have the overhead of TCP.
    - QUIC uses UDP as a transport layer, bypassing TCP's legacy constraints while adding its own mechanisms for reliability, congestion control, and encryption.
- **What QUIC Adds:**
    - QUIC essentially "replaces" TCP's functionality on top of UDP, handling everything (connection management, retransmission, flow control) at the application layer.

---

2. Stream Multiplexing

- **What It Does:**

    - Allows multiple streams (independent sub-connections) within a single QUIC connection.
    - Each stream is independent, meaning data delivery on one stream doesn't block others.
- **How It's Achieved:**
    - Streams are identified by unique IDs.
    - A single QUIC connection can manage multiple streams simultaneously, avoiding TCP's **head-of-line blocking** problem.

---

3. Connection Handshake

- **TLS 1.3 Integration:**
    - QUIC integrates encryption directly into the protocol, combining the transport handshake and encryption handshake into one.
    - This drastically reduces the number of round trips needed:
        - **New connections:** 1-RTT (one round trip).
        - **Resumed connections:** 0-RTT (no round trip).
- **How It Works Internally:**
    - During the handshake, the client and server exchange cryptographic keys and establish the connection in one step.

---

4. Connection Identifiers (CIDs)

- **What It Solves:**
    - In TCP, connections are tied to the IP address and port. If the network changes (e.g., switching from Wi-Fi to 4G), the connection breaks.
- **How It Works Internally:**
    - QUIC uses **Connection Identifiers (CIDs)**:
        - A unique ID is assigned to each connection, decoupling it from the underlying IP/port.
        - Even if the client's network changes, the CID ensures the connection persists.

---

5. Congestion Control and Flow Control

- **Congestion Control:**
    - QUIC incorporates advanced congestion control algorithms, similar to TCP (e.g., Reno, Cubic, or BBR).
    - Internally, QUIC adapts based on network conditions like packet loss, jitter, and delay.

    - It’s modular, meaning developers can implement custom algorithms for specific applications.
- **Flow Control:**
    - QUIC uses per-stream and connection-level flow control to manage the rate of data transmission:
        - Per-stream flow control ensures streams don’t overwhelm the receiver.
        - Connection-level flow control ensures the entire connection remains efficient.

---

6. Packet Structure

- **Compact and Flexible Packets:**
    - QUIC packets are designed to be lightweight and self-contained.
    - A QUIC packet includes:
        - **Header:** Contains the Connection ID, packet number, and flags.
        - **Payload:** Data for streams, frames, or control information.
    - Packets are encrypted end-to-end, ensuring confidentiality and integrity.
- **Packet Numbers vs. Sequence Numbers:**
    - Unlike TCP, QUIC uses packet numbers, not sequence numbers.
    - Packet numbers are unique and not affected by retransmissions, simplifying loss detection.

---

7. Error Detection and Recovery

- **Forward Error Recovery:**
    - QUIC quickly detects lost packets without relying on retransmission timeouts like TCP.
    - Uses acknowledgment (ACK) frames to inform the sender about received or missing packets.
- **Efficient Retransmission:**
    - Only the lost packets are retransmitted, without affecting other streams in the connection.

---

8. Encryption and Security

- **TLS 1.3 Integration:**
    - QUIC includes encryption by default, making all connections secure.
    - Encryption protects:
        - Connection metadata (e.g., packet numbers).
        - Payload data (e.g., trading information or API responses).

- **Perfect Forward Secrecy (PFS):**
    - Each session uses unique keys, ensuring past communications remain secure even if future keys are compromised.

---

9. Extensibility

- **Frame-Based Design:**
    - QUIC uses frames to define its functionality. Each frame type represents a specific action (e.g., data transmission, acknowledgments, or stream management).
    - This modular approach allows QUIC to be extended for future needs. For example:
        - Adding support for new congestion control algorithms.
        - Enhancing real-time streaming capabilities.

Internals vs. TCP

Aspect	TCP	QUIC
Transport Layer	Operates at Layer 4 (built-in OS).	Operates at the application layer.
Encryption	Optional (via TLS).	Mandatory (TLS 1.3 by default).
Packet Handling	Sequence numbers for streams.	Packet numbers independent of streams.
Stream Multiplexing	No, head-of-line blocking occurs.	Yes, streams are independent.
Network Migration	Not supported.	Supported with Connection IDs.
Reliability	Retransmissions delay all data.	Stream-specific retransmissions.

How QUIC Fits into Fixed Income Banking

1. **Low Latency Execution**:
    - The use of UDP, faster handshakes, and independent streams ensures fast trade execution.
2. **Resilient Data Feeds**:

    - Advanced loss recovery ensures market data (e.g., bond prices, yield curves) remains real-time, even during network instability.
3. **Streamlined API Responses**:
    - QUIC's frame-based architecture improves the speed and reliability of analytics and pricing APIs for institutional clients.
4. **Seamless Mobility**:
    - With CIDs, traders switching networks (e.g., Wi-Fi to mobile) stay connected to trading platforms without disruptions.