What transport problems does QUIC solve and how does it do it?

The main problems with TCP:

1. Latency because of costly connection establishment
2. Head-of-line blocking
3. Loss detection

How does QUIC do it?

1. **Solving the latency problem**

   A TCP + TLS connection (what is generally used, as of now, for internet transport) establishment requires at-most 3-RTTs. A QUIC connection, on the other hand, can do a connection establishment in 0-RTT in most cases where the client has communicated with the server before. In a worst case scenario, QUIC can do it in 1-RTT.
   QUIC combines the TCP + TLS handshake by exchanging encryption information and connection establishment in a single handshake resulting in at-most 1-RTT.
   Majority of the latency issues are solved just by this step.

2. **Solving head-of-line blocking issue**

   TCP is like a sequential bit pipe. All objects are put in the pipe and a loss of a packet in one object makes all subsequent objects wait until the lost packet is retransmitted and received. QUIC overcomes this problem by using stream multiplexing, where each object has an independent stream and a packet loss only halts that particular stream and other streams can continue transmission.

3. **Better loss detection**

   A packet loss in TCP might result in retransmission ambiguity. Retransmission ambiguity occurs when a sender has retransmitted packet and receives ACK for the data. The sender cannot be certain whether the ACK received is for the original packet or the resent packet. In case of QUIC, when a packet is deemed lost, QUIC bundles a new packet with a new packet number, removing any ambiguity about which packet is received. This helps in detecting spurious retransmissions easily. This is why QUIC outperforms TCP in scenarios with fluctuating bandwidth.