Connection Migration

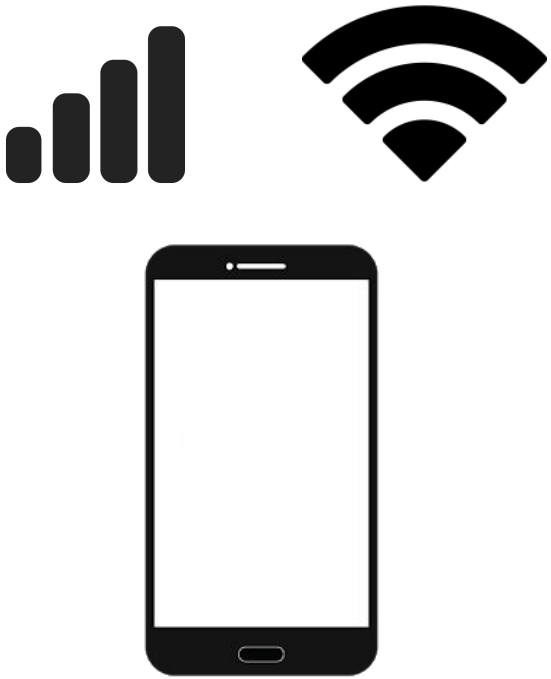
Cherie Shi, Google

IETF 104, Prague
March, 2019

Connection Migration

Seamlessly migrate unfinished requests **between** different network interfaces.

"**Parking-Lot** Problem"



A Quic(k) Demo


No Connection Migration in QUIC
With Connection Migration in QUIC

Connection Migration Signals

signals that could lead to a connection migration attempt

- **Platform Notifications**
  - OnNetworkDisconnected
  - OnNetworkMadeDefault
  - OnNetworkConnected, etc
- **Write Errors**

  preemptive signals of network change
- **Path Degrading Detections**

Data

Opportunity Size

At the **application** level: **2%** requests failed with **NETWORK_CHANGED**

Data

Opportunity Size

At the **application** level: **2%** requests failed with **NETWORK_CHANGED**

At the **connection** level:

- **7.87%** connections are closed due to **NETWORK_CHANGED**
- **0.72%** connections encounters preemptive **PACKET_WRITE_ERROR**s

  caused by network changes.

In total, **8.59% connections** MAY be subject to migrate

Data

Stage 1: On Platform Notification & Write Error

- Text search: **-0.7% failures, -0.3% cancels**

Confidence Level: 99%

Data

Stage 1: On Platform Notification & Write Error

Application level

- Text search: **-0.7% failures, -0.3% cancels**

Confidence Level: 99%

BUT... 2% requests may be subject to the feature...

Connection level

- On write error signals, **99.04%** have **handshake unconfirmed**;
- Of all migration attempts, **31.41%** has **no alternate network**;
- Some connections **detect path degrading** before platform notification.

Data

Cont'd: opportunity size at the session level

- **1.10%** detect path degrading then a platform notification
- **5.63%** connections fail with **handshake timeout**

Data

Cont'd: opportunity size at the session level

- **1.10%** detect path degrading then a platform notification

➜ **Trigger migration on path degrading**

- **5.63%** connections fail with **handshake timeout**

➜ **Solve before handshake cases**

Data

Stage 2: On Path Degrading & Before Handshake

- Search **latency** - server response time
  - Text: **-1.25% overall**

    -1.47% at 50%tile, -1.35% at 90%tile, -0.69% at 99%tile
  - Voice: **-0.52% overall**

    -1.25% at 50%tile
- Text search: **-1.4% failures, -1.9% cancels**

Two Principles

Principle I

do not fail the request if it could succeed

Principle II

respect the platform's choice of default network

Design Idea

Handshake confirmed

- **Immediate migration** - jumps away with no testing
- **Migration after probing** - migrates with confidence

Handshake not confirmed

Spin up a **new** connection on the **alternate** network

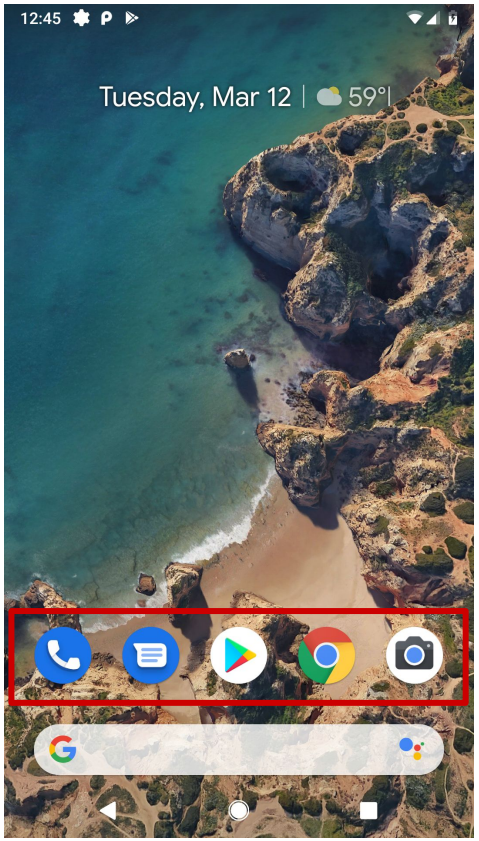
Thank you!

Migration Handling

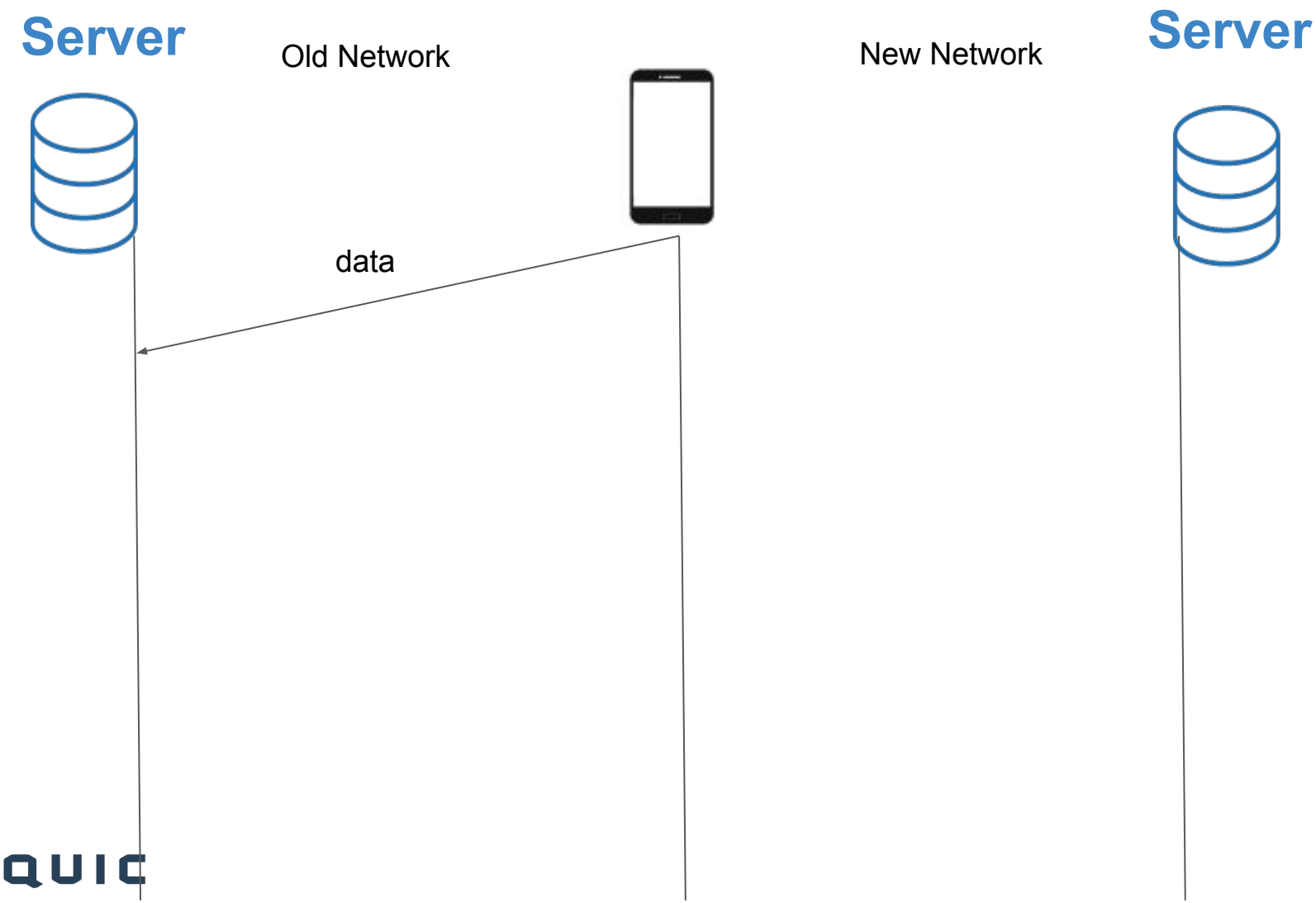
- **Probe** if there is **at least one** possibly working network
  - Current network is degrading but still up
  - A new network is marked as default, current network is still up
- Migrate **immediately** if there is **at most one** working network
- When on the **non-default** network, **periodically probe** the **default** network and move back if it is working until
  - Platform changes default
  - Successfully migrate back to default

The Demo App: Android Google Search

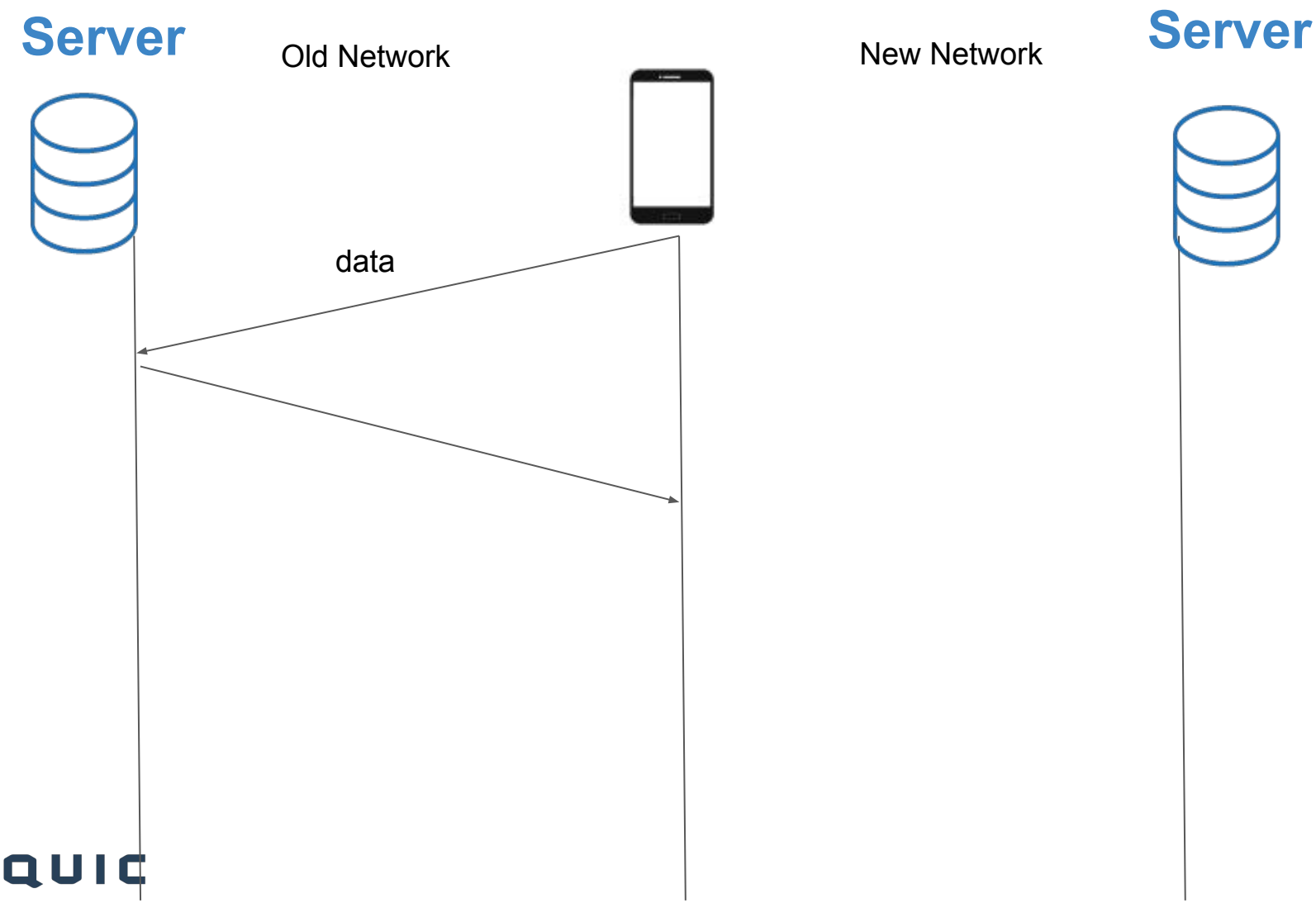
- Users send requests, expect response
  - Sent to www.google.com
  - Text or voice search
  - Demo used voice search
- Mobile users are usually **on the move**
  - Subject to network changes



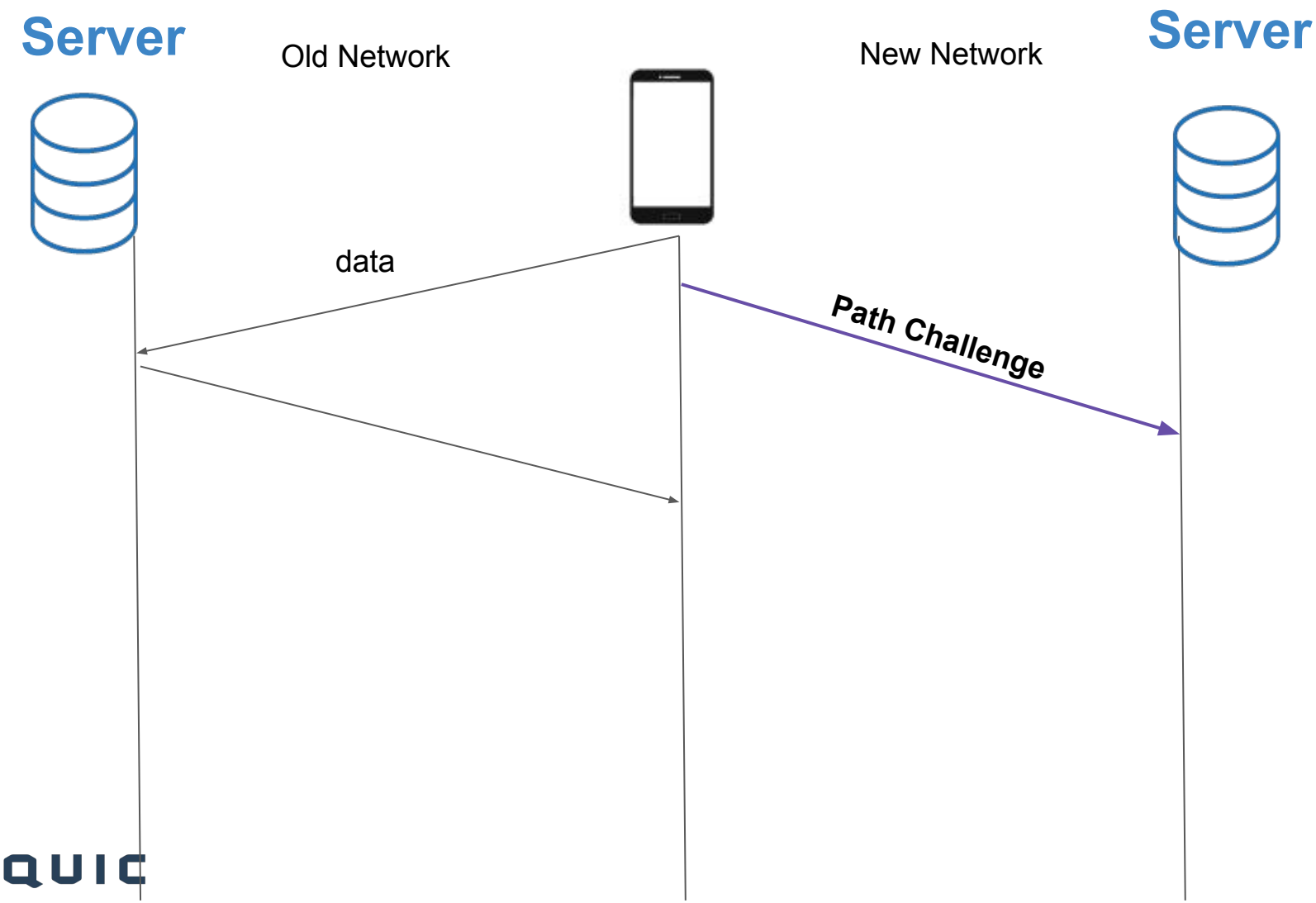
Server and Client Interaction



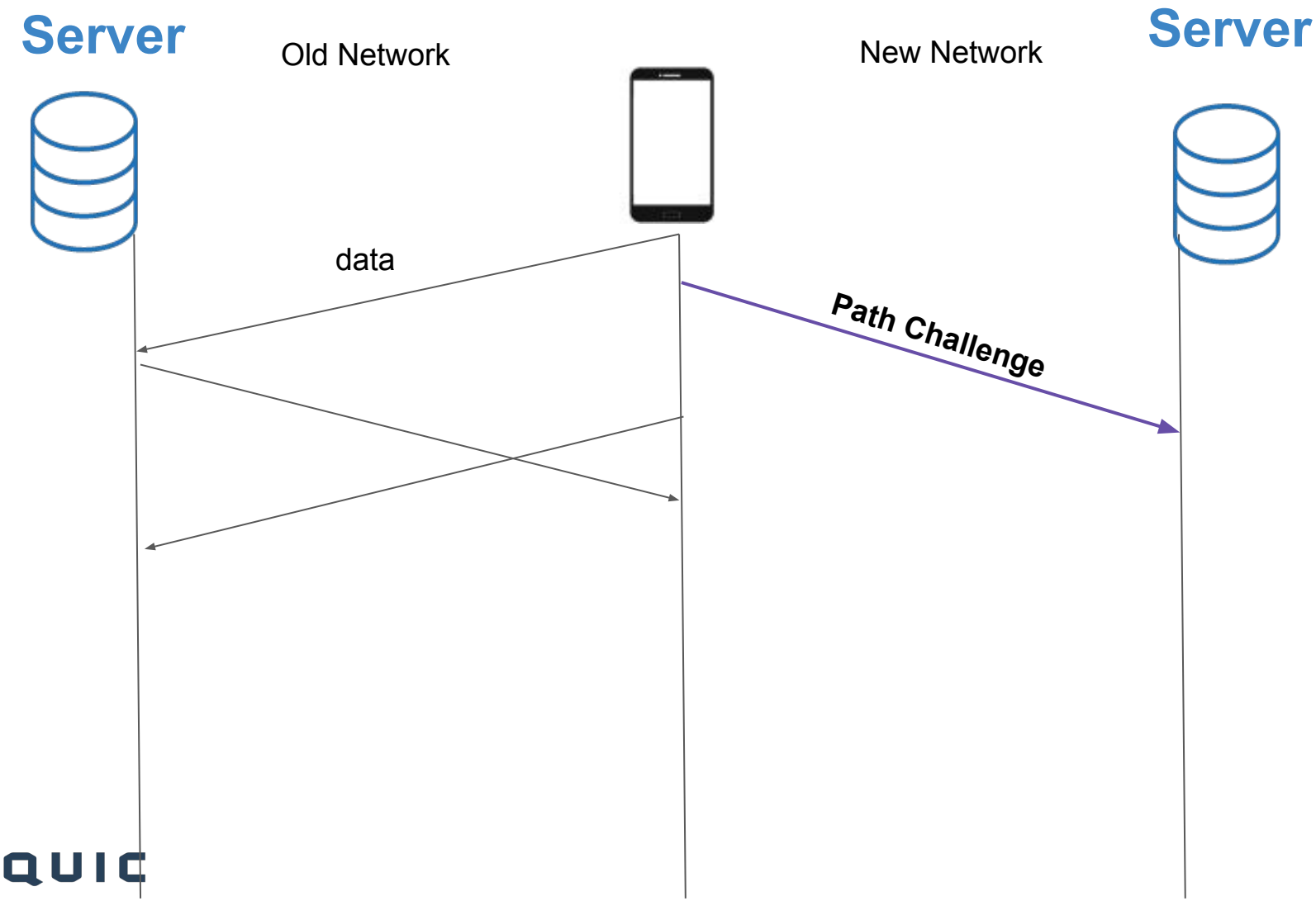
Server and Client Interaction



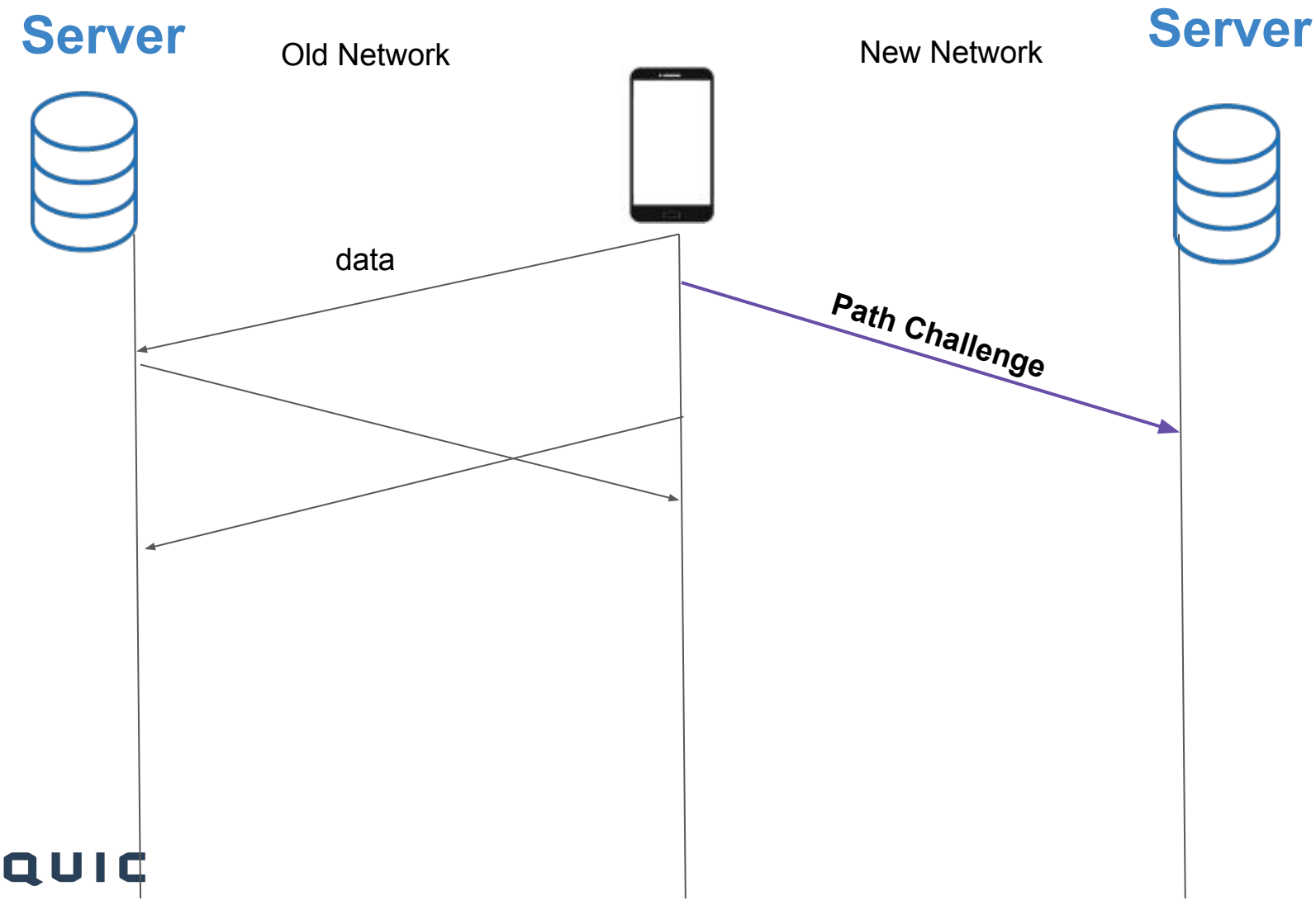
Server and Client Interaction



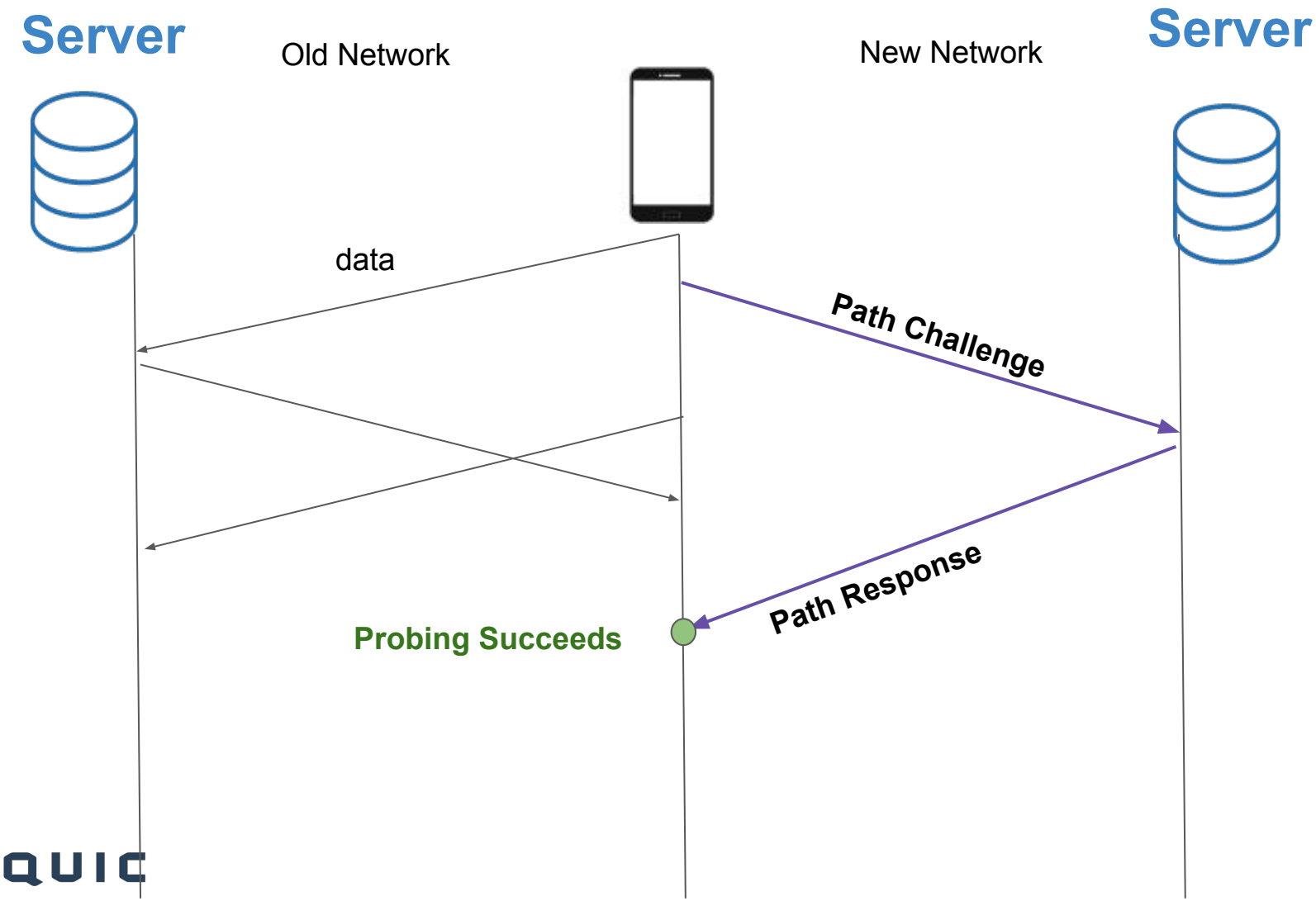
Server and Client Interaction



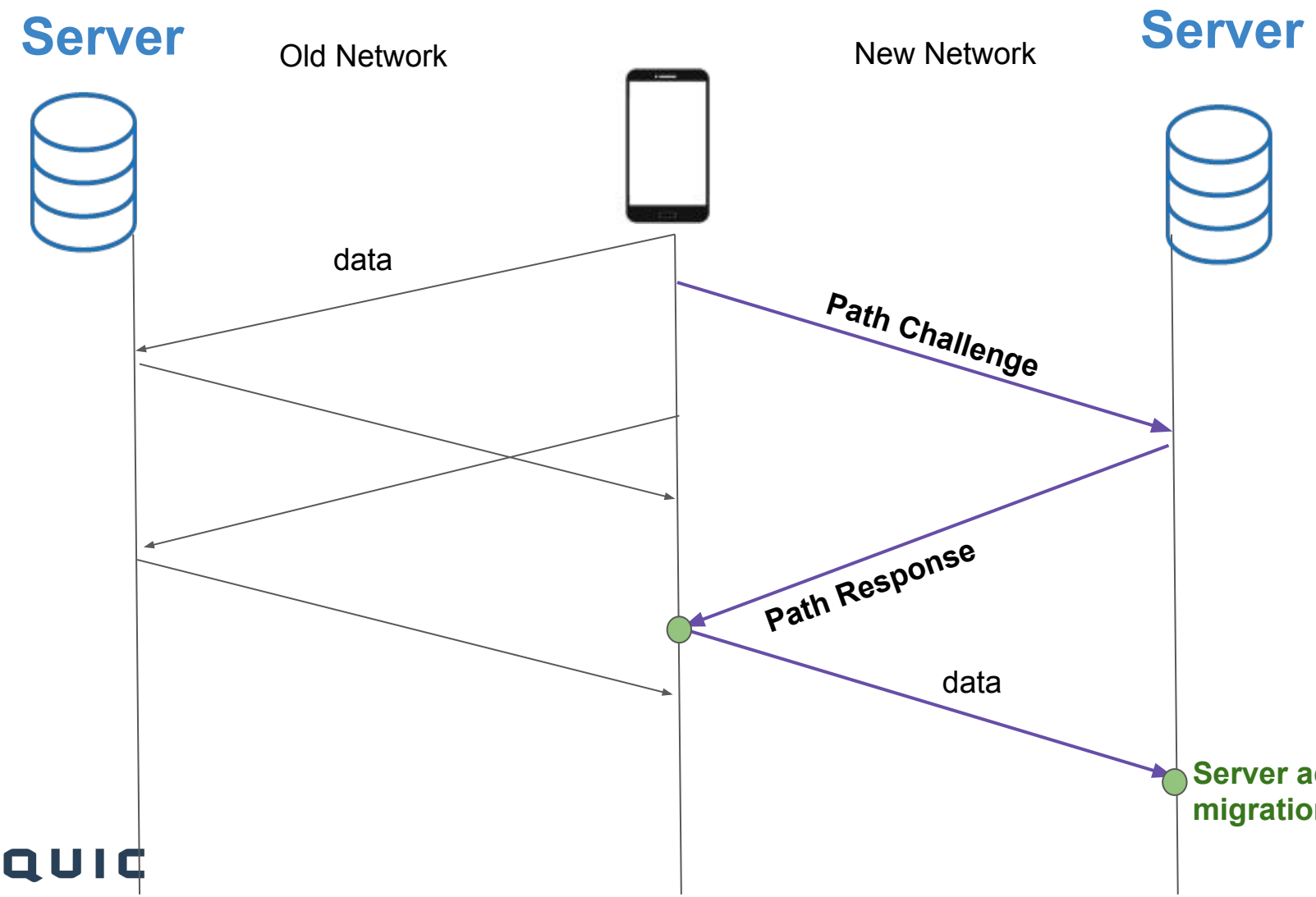
Server and Client Interaction



Server and Client Interaction



Server and Client Interaction



Server and Client Interaction