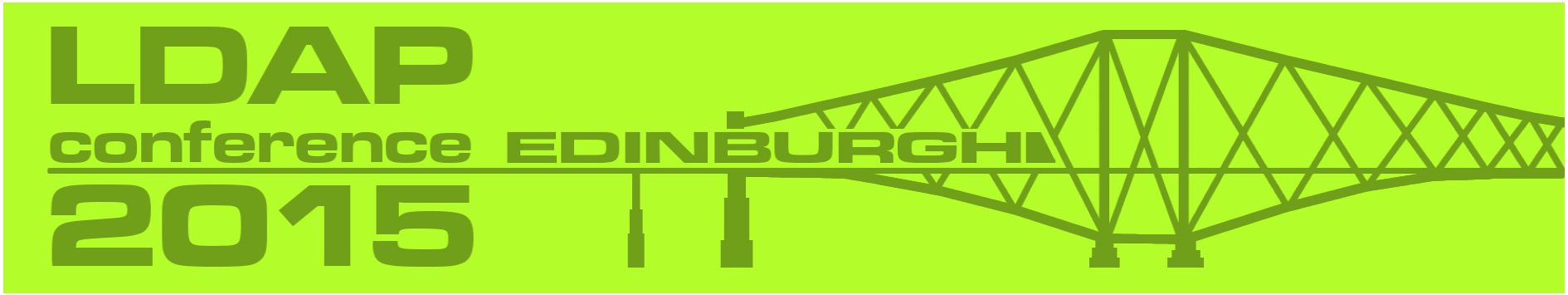


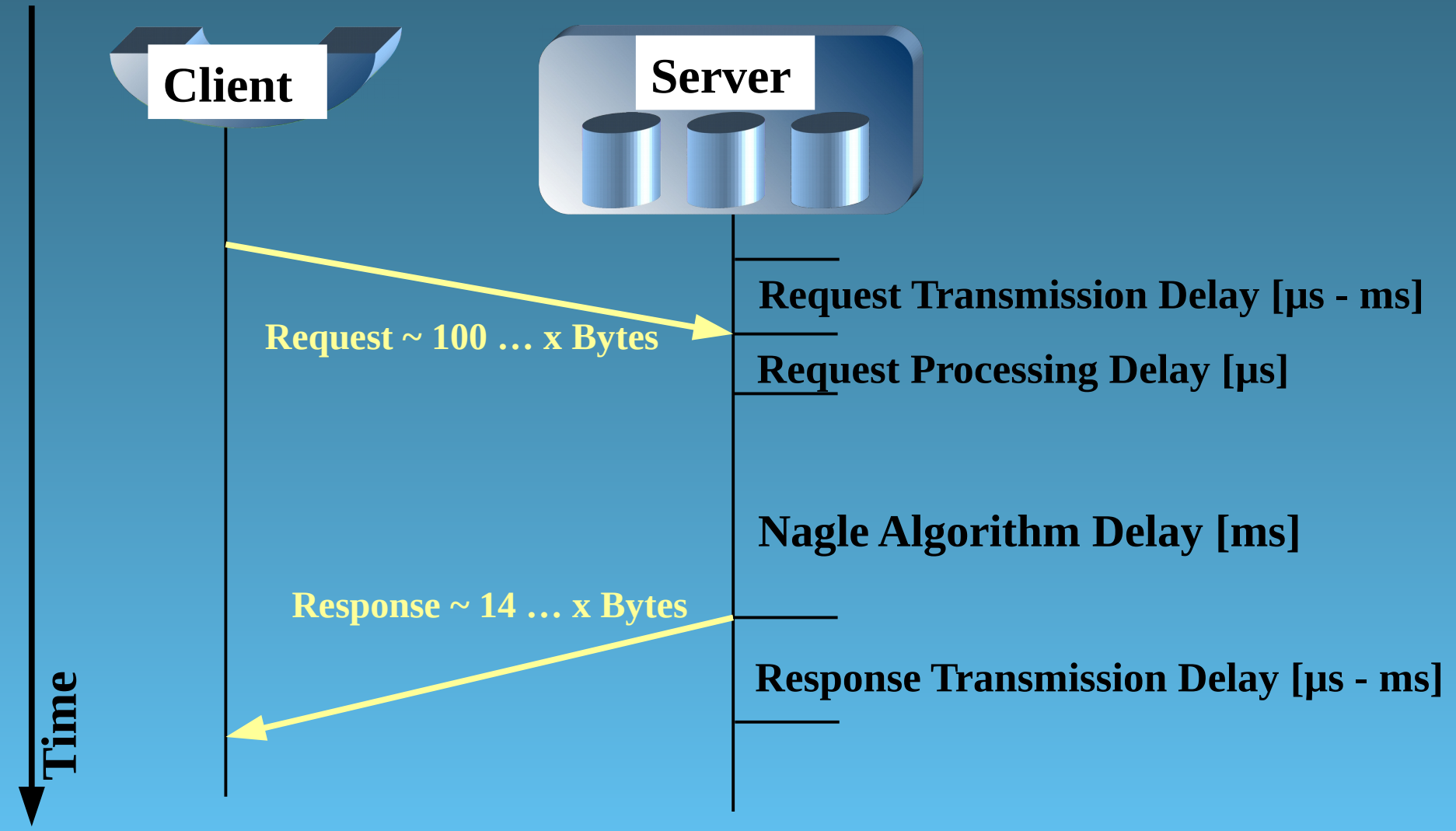
Improvements of the LDAP Protocol

Christian Hollstein, TeraCortex

www.teracortex.com

- **Asynchronous Queue Length Control**
- **Transaction Extensions and Controls**
- **Performance: Experimental Results**

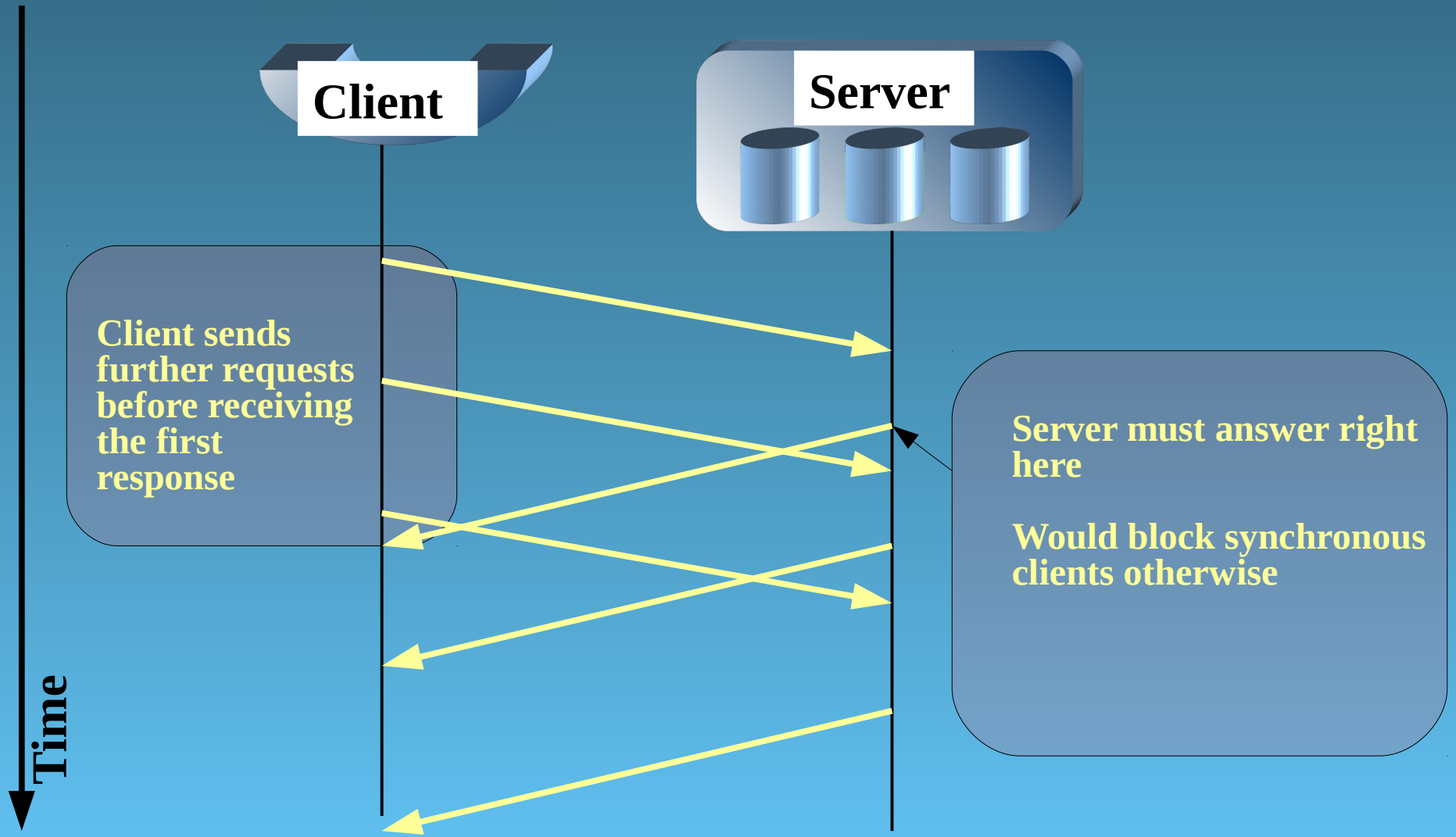
Synchronous LDAP Protocol Sequence



Properties of Synchronous Protocol Pattern

- **Many LDAP requests and all update responses have bad TCP packet utilization (1% - 30% of MTU ~ 1500 Bytes)**
- **Nagle algorithm improves TCP efficiency at the expense of increased latency**
- **Asynchronous mode increases TCP efficiency for requests but not for responses**

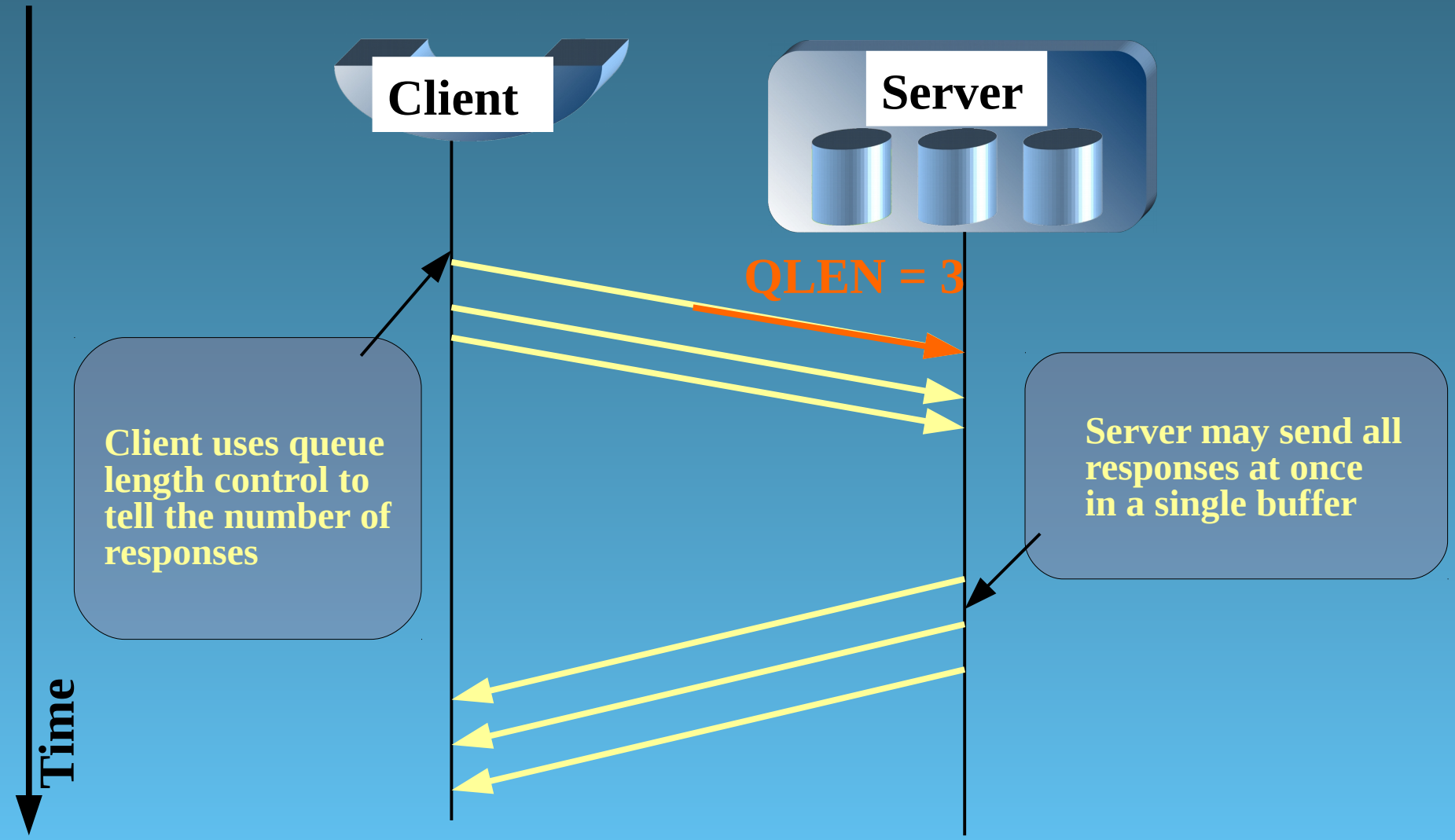
Asynchronous LDAP Protocol Sequence



Properties of Asynchronous Protocol Pattern

- Good TCP packet utilization for requests
- Response pattern still inefficient due to immediately required server reaction
- No LDAP protocol element to let the client tell the server the number of expected responses

Solution: Attach LDAP Queue Length Control



Properties of Optimized Asynchronous Protocol

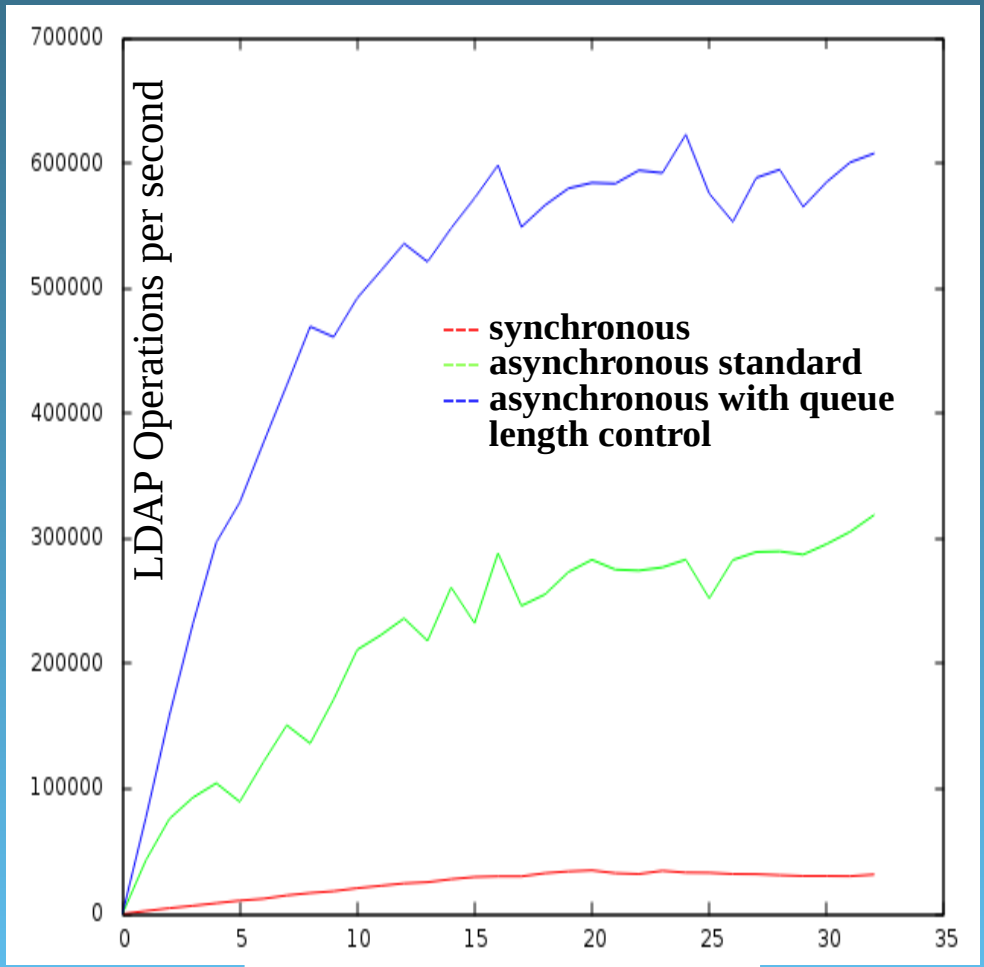
- **Best TCP packet utilization for requests *and* responses**
- **Both sides may disable Nagle algorithm**
- **Combines minimum response times with maximum TCP throughput**
- **LDAP queue length control specification available as IETF draft**

Test Arrangement in Amazon EC2 Cloud

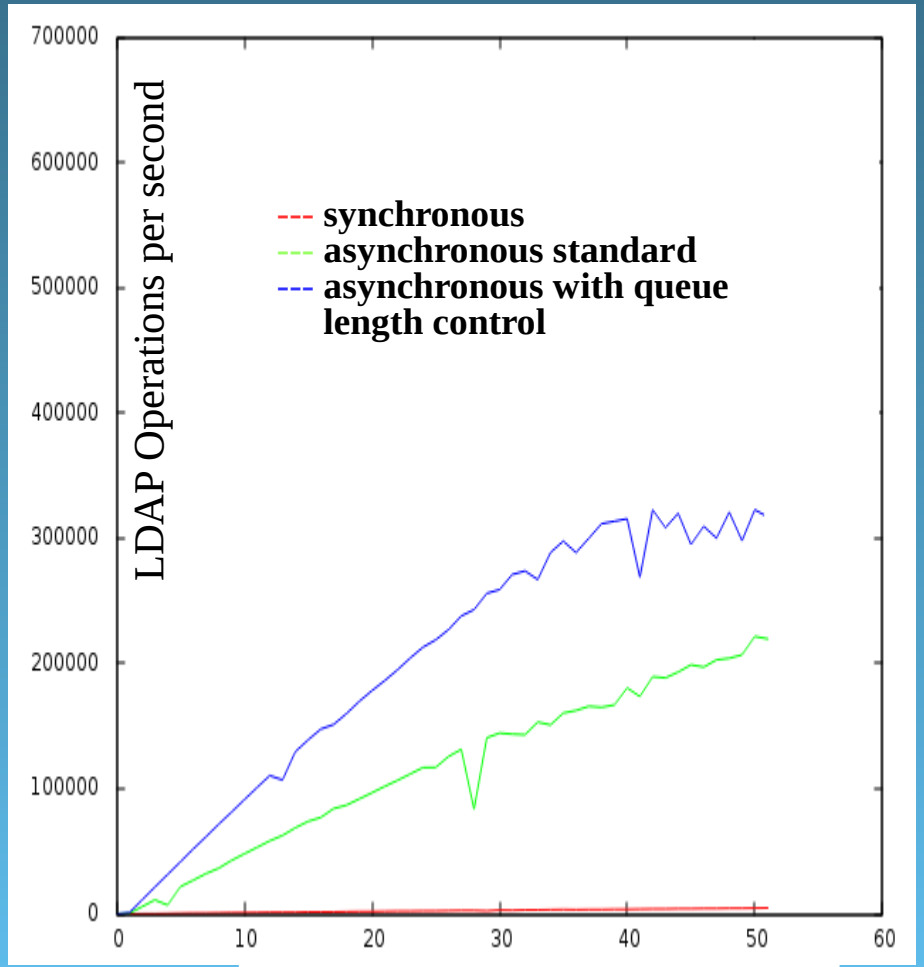
- m4.2xlarge instances: 2 x Client, 1 x Server
- Sites near Washington and Frankfurt
- US client and server in same data center
- SLES 12
- DVTDS 3.2.2 LDAP Server, 32 million entries
- ELDC 1.006 load driver

Experimental Results

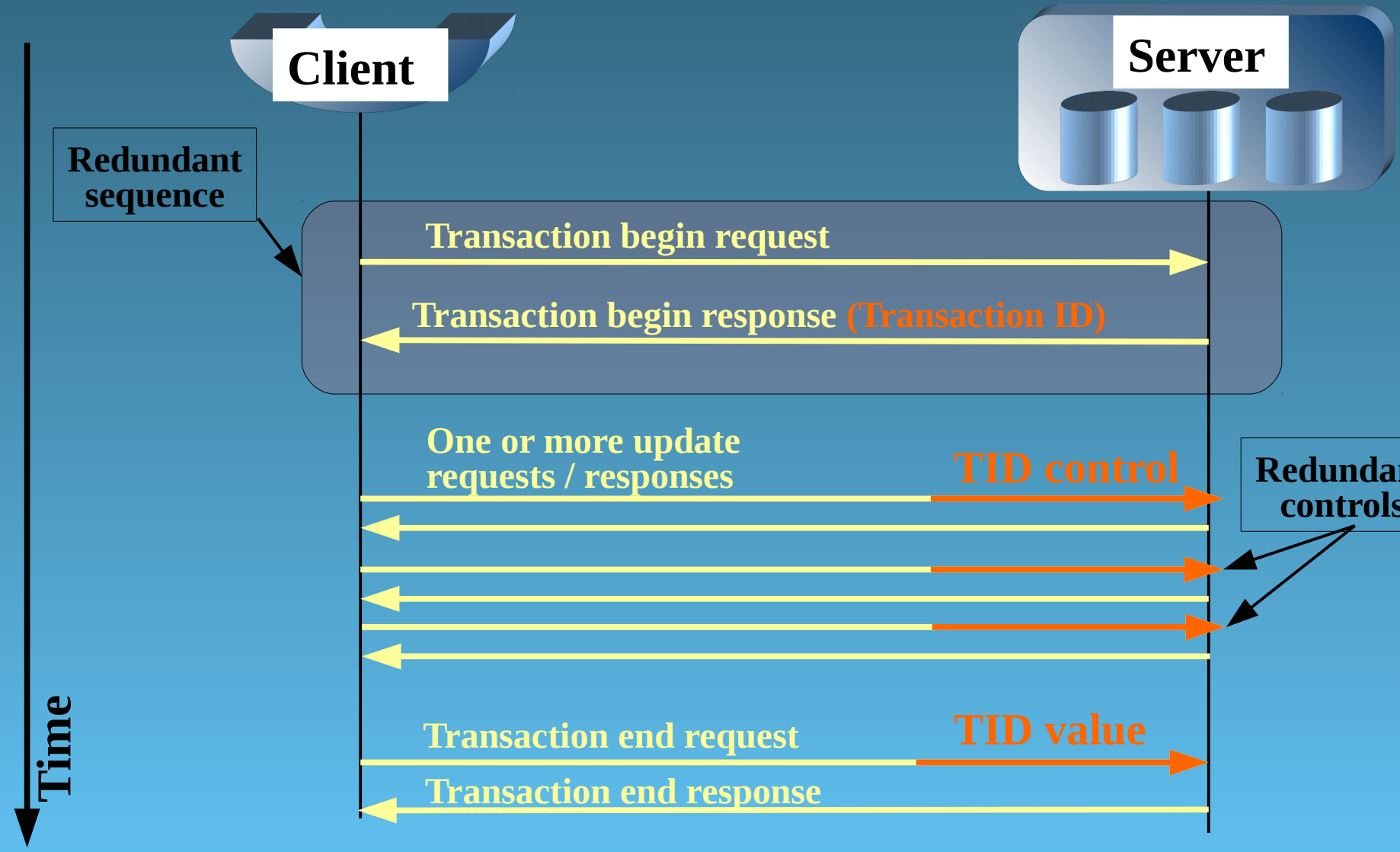
Local throughput, same sub net



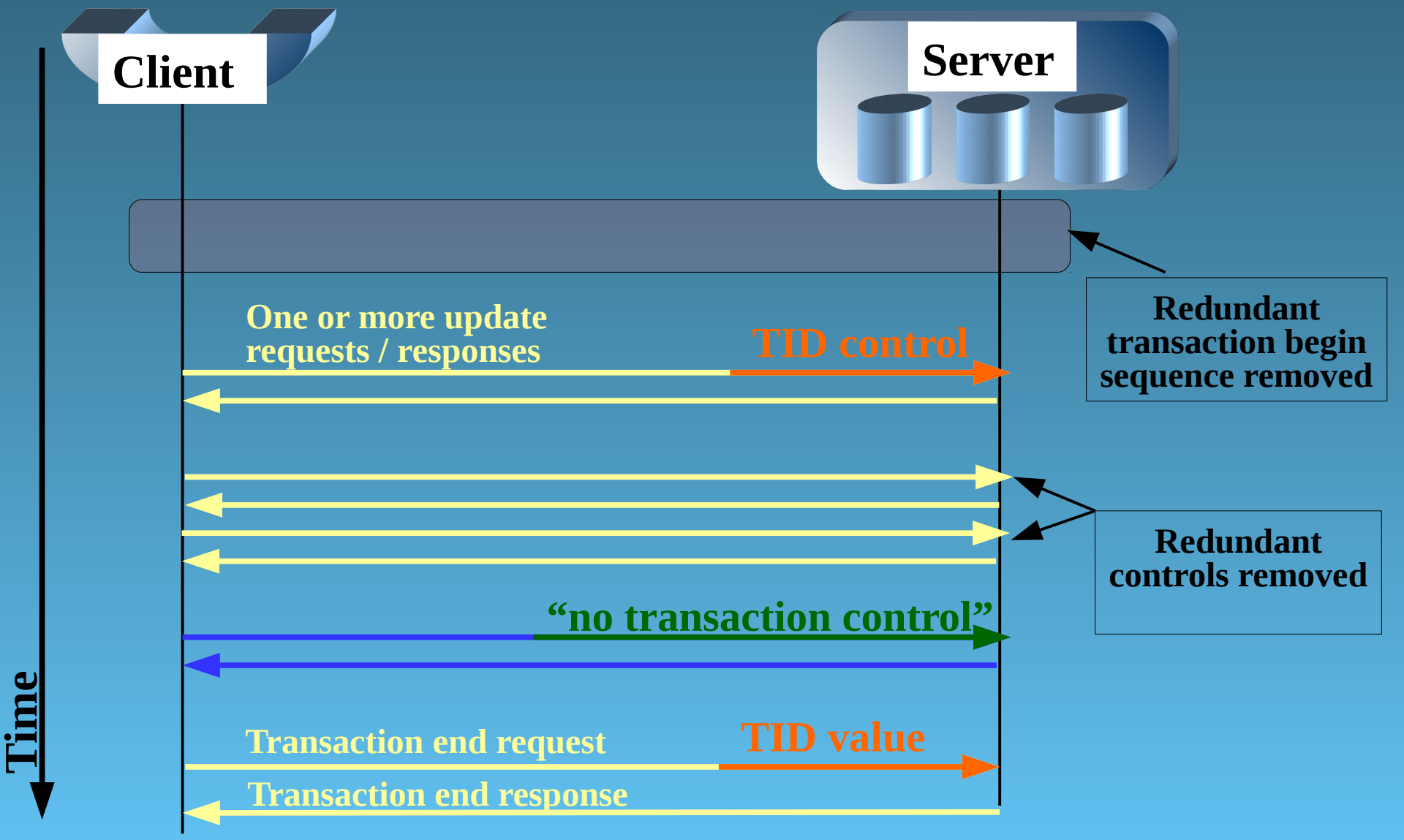
Throughput across Atlantic ocean



LDAP Transaction Protocol Sequence (RFC5805)



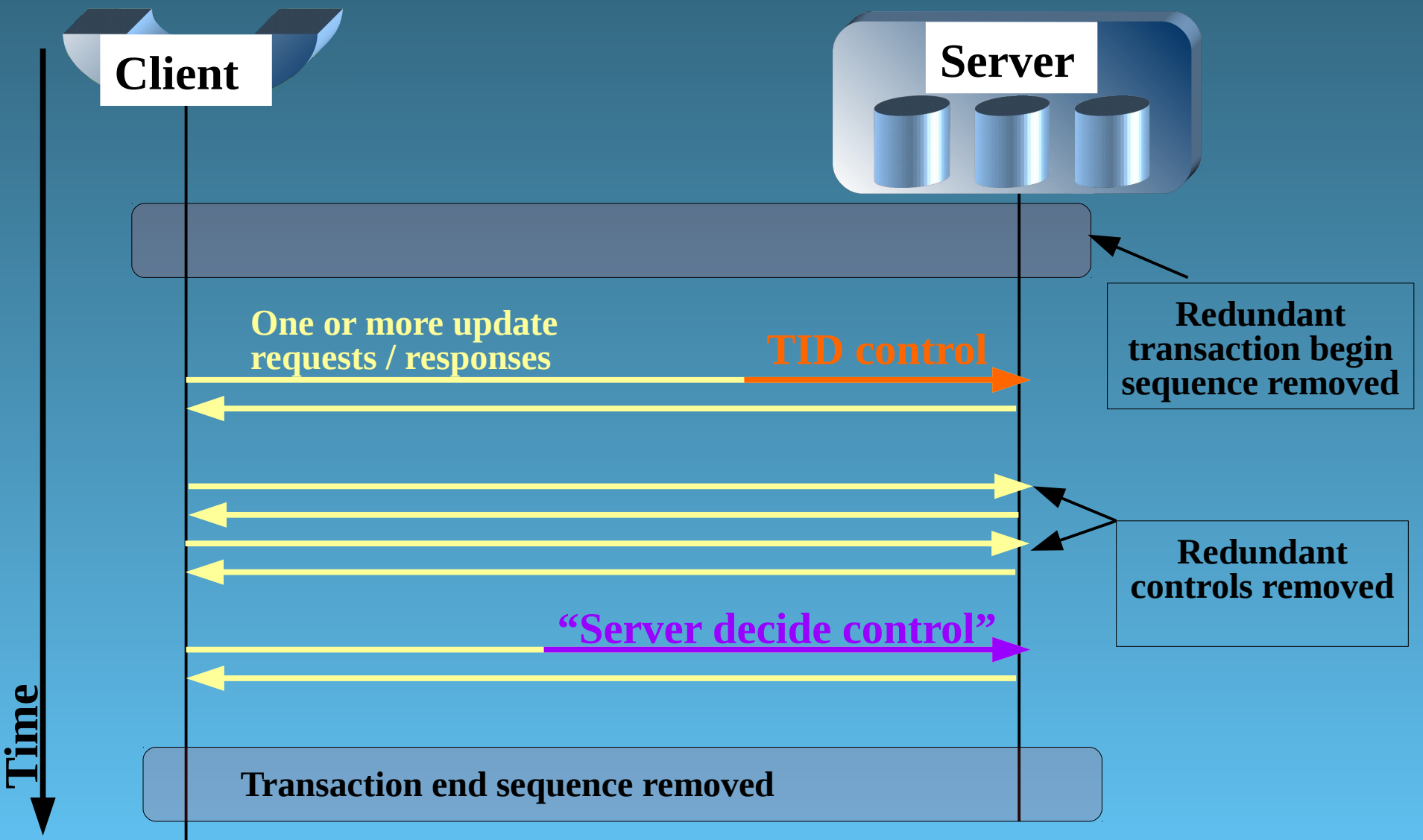
Removed Transaction Begin and Controls



Properties of Optimized Transactions

- **Client may chose the TID. Server starts transaction with the first request carrying the transaction control**
- **Requests outside the transaction carry a *“non transaction control”***
- **Requests inside the transaction don't need redundant transaction control**
- **Avoids latencies of synchronous transaction begin sequence and saves bandwidth**

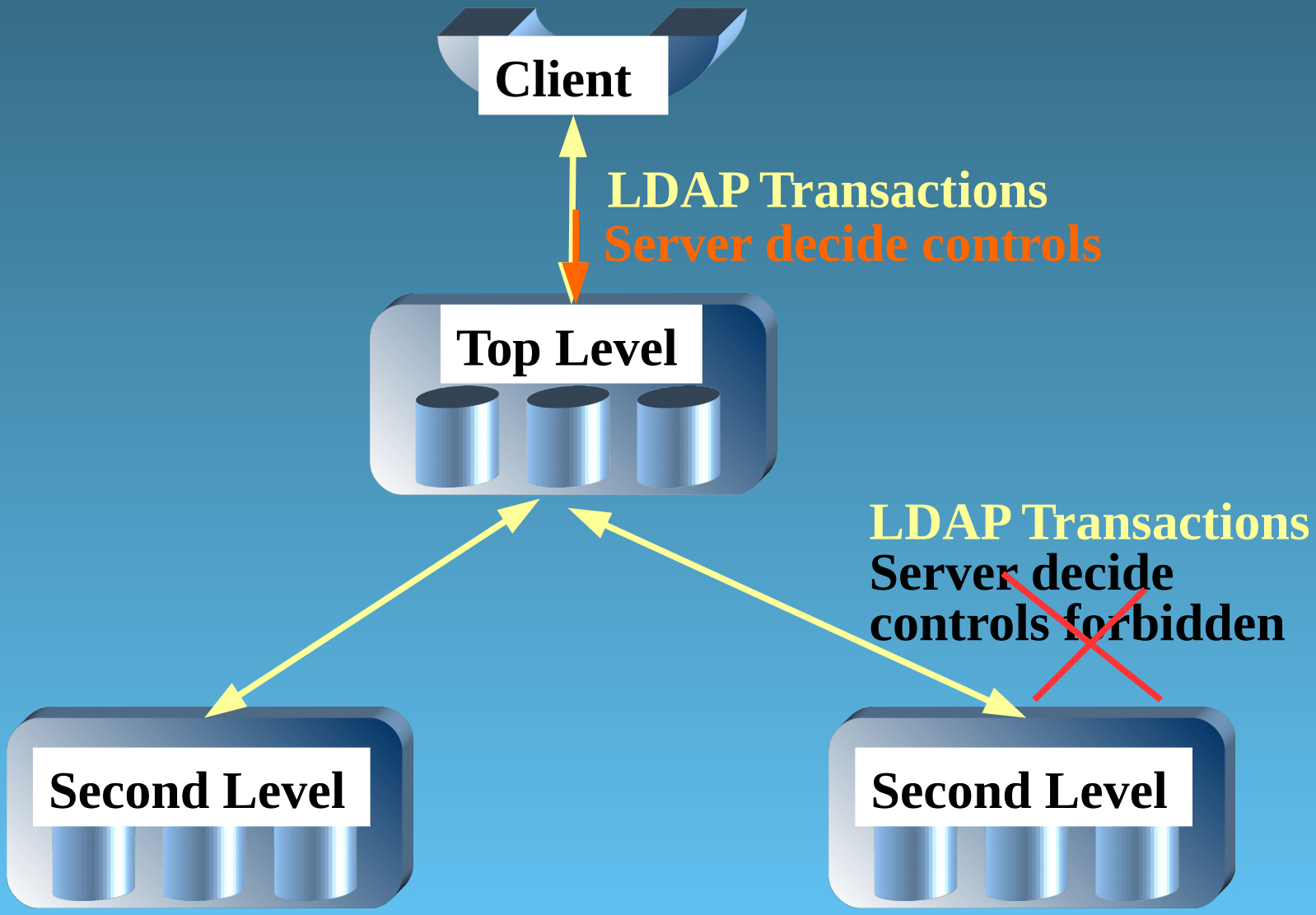
Server decided Commit / Rollback



Properties of Server decided Commit / Rollback

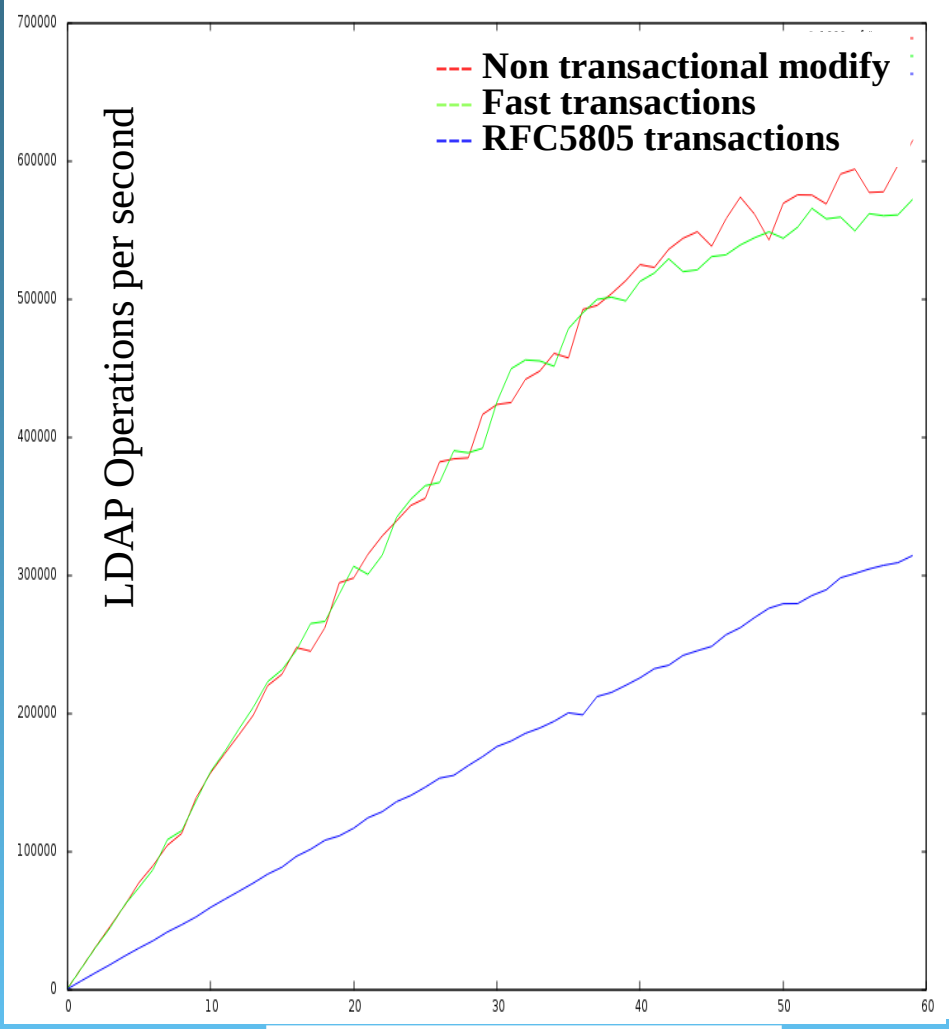
- **Client attaches a “server decision control” to the last request in transaction sequence**
- **Server decides commit / rollback upon outcome of requests**
- **Avoids latencies of synchronous transaction end sequence**
- **Minimized overhead compared to non transactional updates**
- **In distributed directories only allowed for first client access point: Servers *MUST NOT* chain this control**

Pitfall: Distributed Transactions

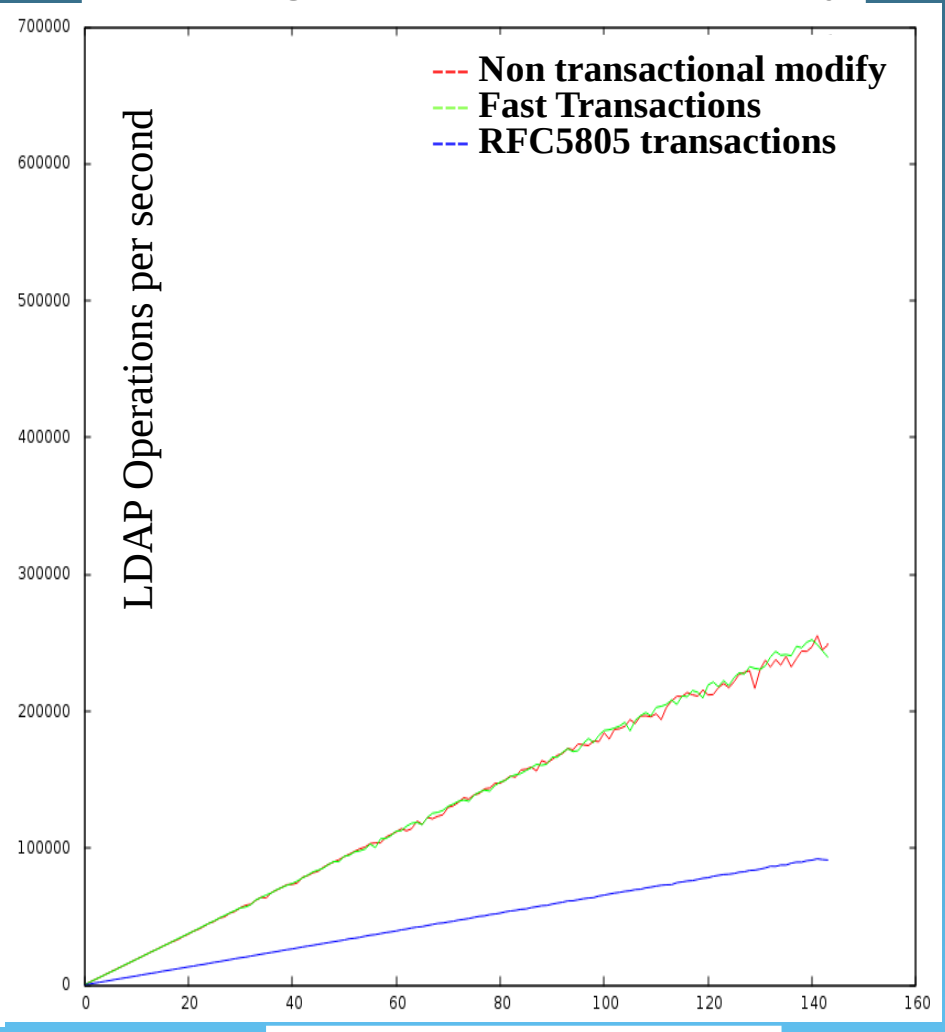


Example: Transaction Throughput over slow Networks

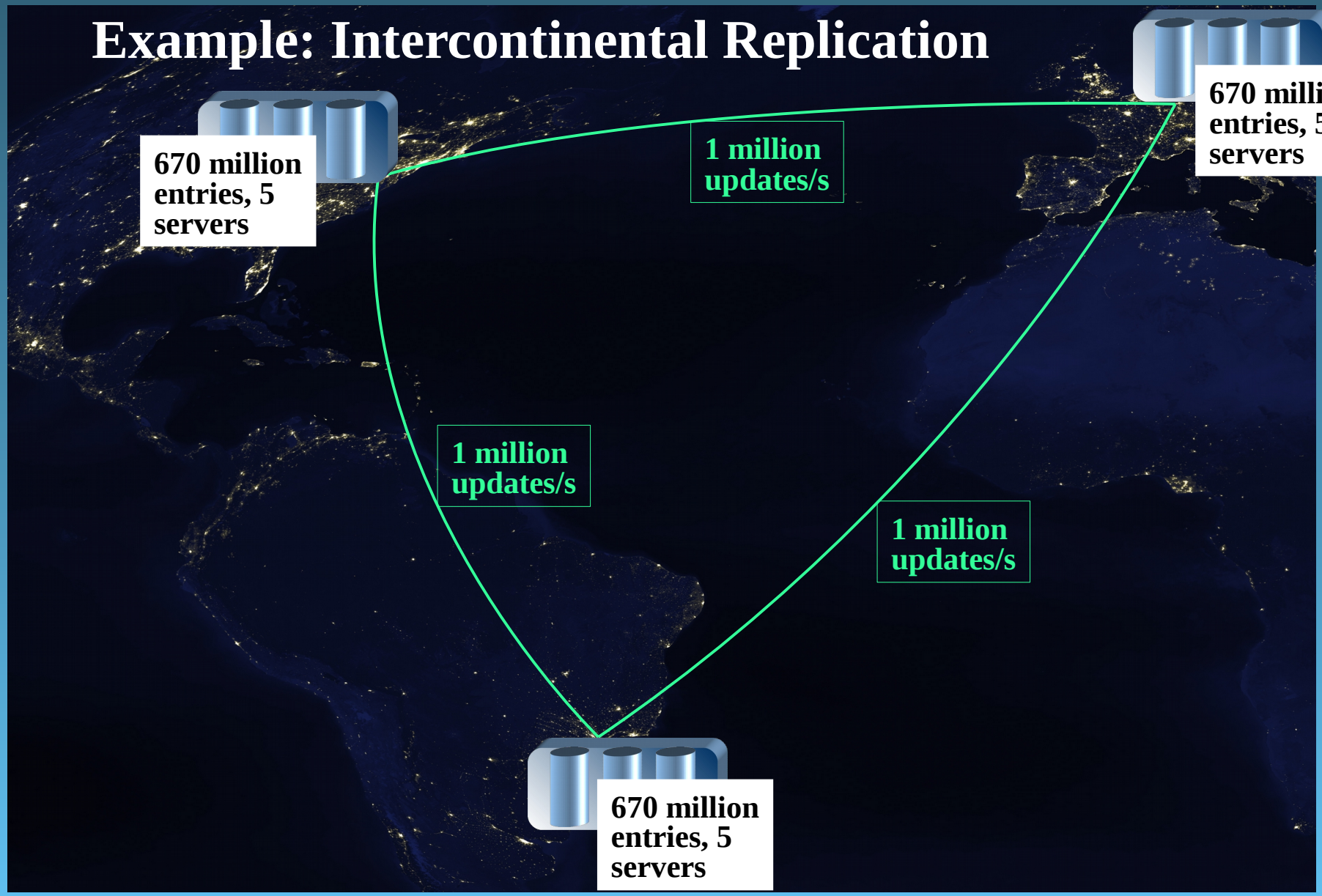
Queue length 1000, 50 ms network latency

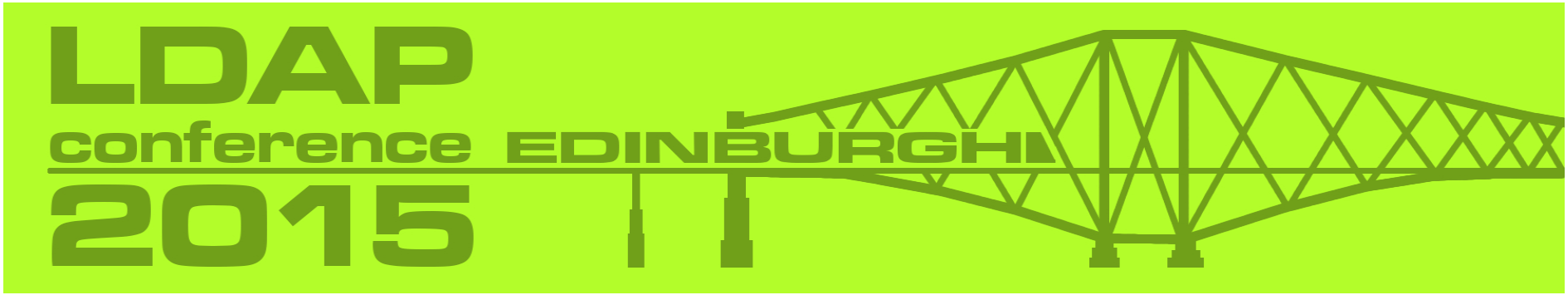


Queue length 100, 50 ms network latency



Example: Intercontinental Replication





Thank you for your attention

www.teracortex.com