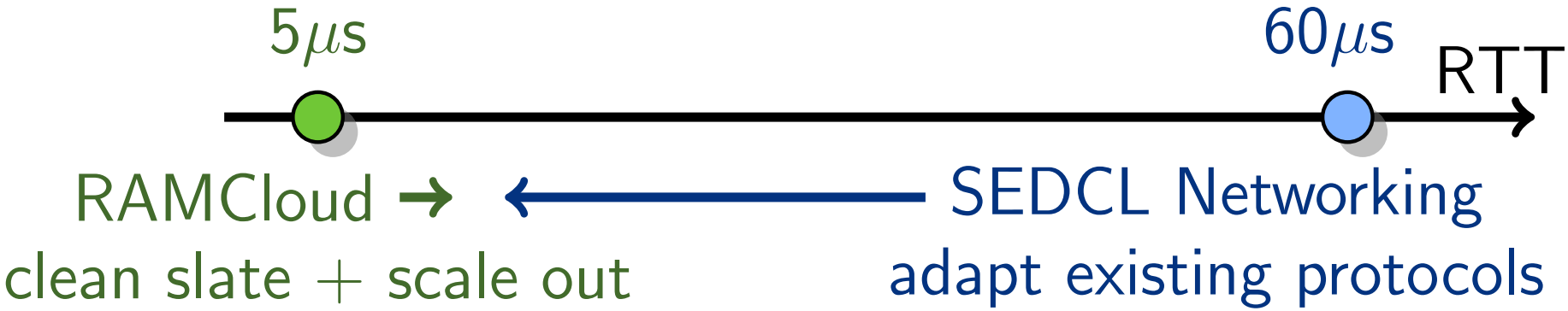
RAMCloud's RPC Protocol

Diego Ongaro

June 4, 2011

Need a New Transport Protocol



Networking research isn't solving our problems yet

- Hint: If you measure in milliseconds, that's not low latency.
- Hesitant to part ways with TCP
  - We can experiment in the datacenter
- Not using the same assumptions

Outline

1. Requirements and assumptions for RAMCloud's RPC System
2. RAMCloud's transport interface: a research platform
3. Key ideas from the FastTransport protocol
4. Results of a simple RPC benchmark

RPC System Requirements

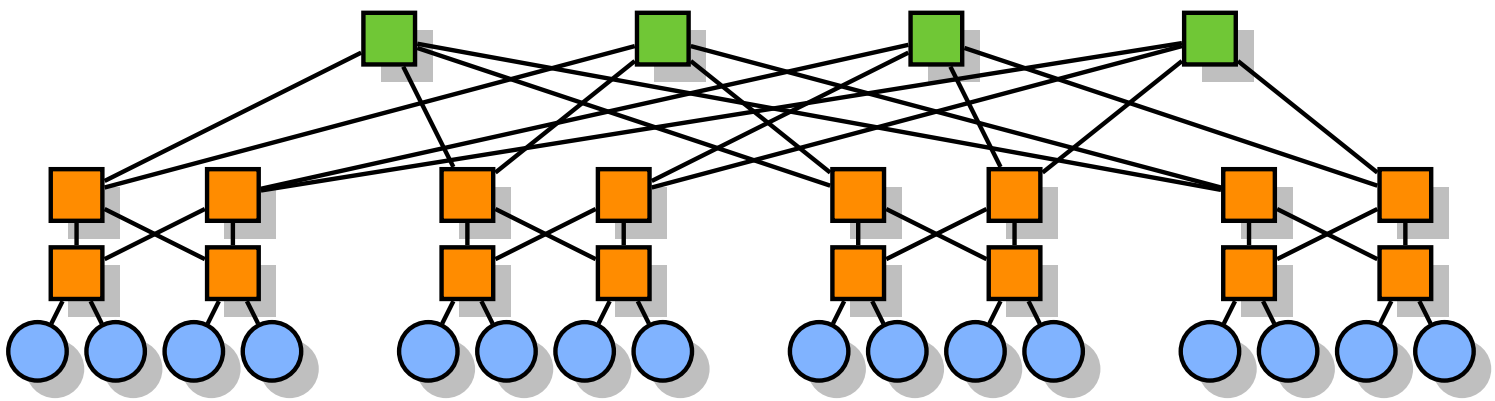
- Low latency (5-10μs)
  - Small reads will dominate workload
- High throughput
  - Larger objects (up to 1 MB)
  - Recovery traffic (up to 8 MB)
- 10,000s of sessions per server
- RPC abstraction
  - Easy to use
  - Asynchronous interface for parallelism

Transport Protocol Assumptions (1/2)

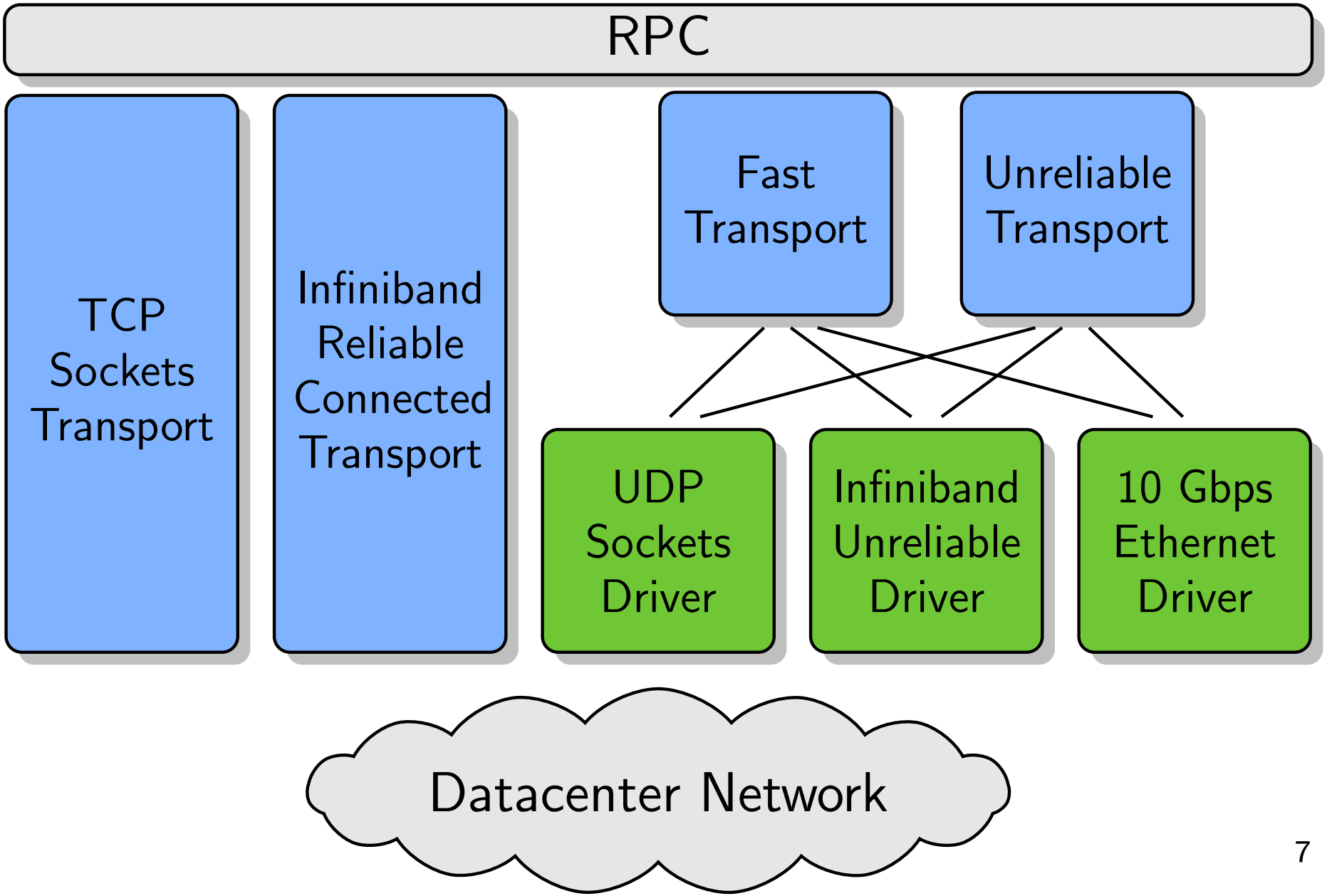
- RPCs
  - Message length is known up front
  - Response acknowledges request
  - Utility in completed RPCs, not bytes sent
  - Traditionally, streaming protocols
- Dedicate a core
  - Poll for packets, TSC for timeouts
  - No delays, fast retransmissions
  - Traditionally, interrupts, clocks, syscalls

Transport Protocol Assumptions (2/2)

- Simple flow control
  - Small windows fill the pipe
  - End hosts have sufficient buffers
  - Traditionally, long links and slow recipients
- Multipath fat tree topologies
  - Full bisection bandwidth
  - Must tolerate packet-level reordering
  - Traditionally, single path with no reordering

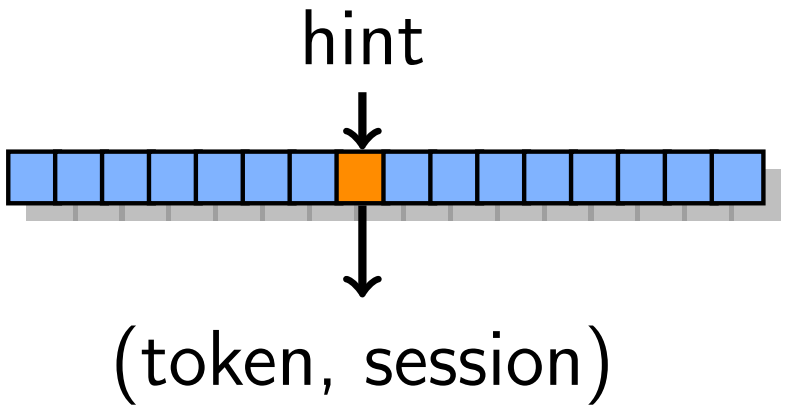


Pluggable Transports



Fast Session Lookup

- 10,000s of sessions
- Lookup state when a packet arrives
- Use hint as index into session table



- Use token to verify hint is still valid

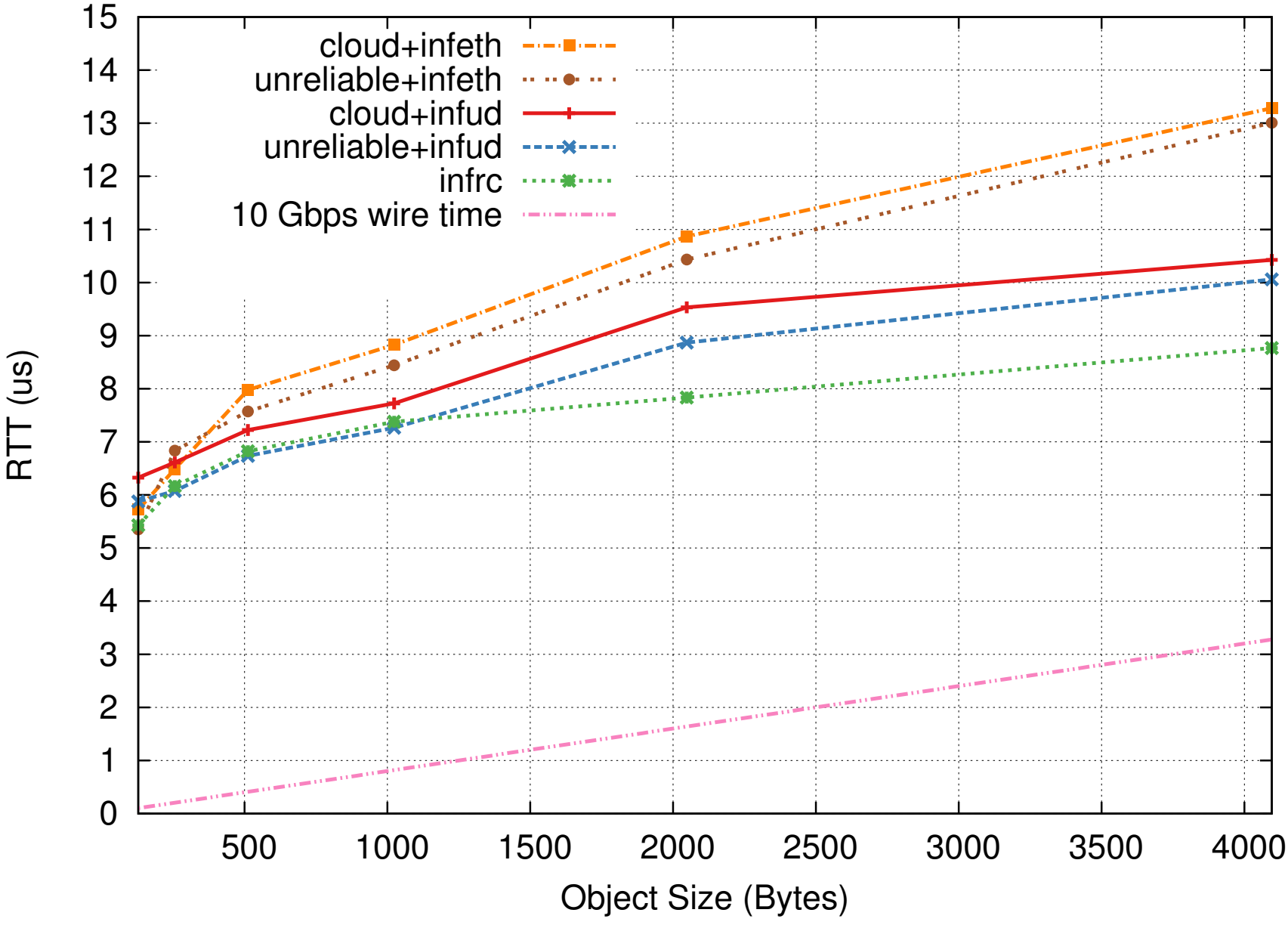
Packet-Level Reordering

- In a multipath network, packets can arrive out of order
- Challenging for TCP
  - Cumulative ACKs
  - ACK bytes, not packets
  - Large receive windows with high delay links
- Addressed in TCP-SACK: list of byte ranges
- Simpler (faster) in FastTransport
  - ACK on the packet level
  - First fragment not yet received and fixed size bit vector for remaining window

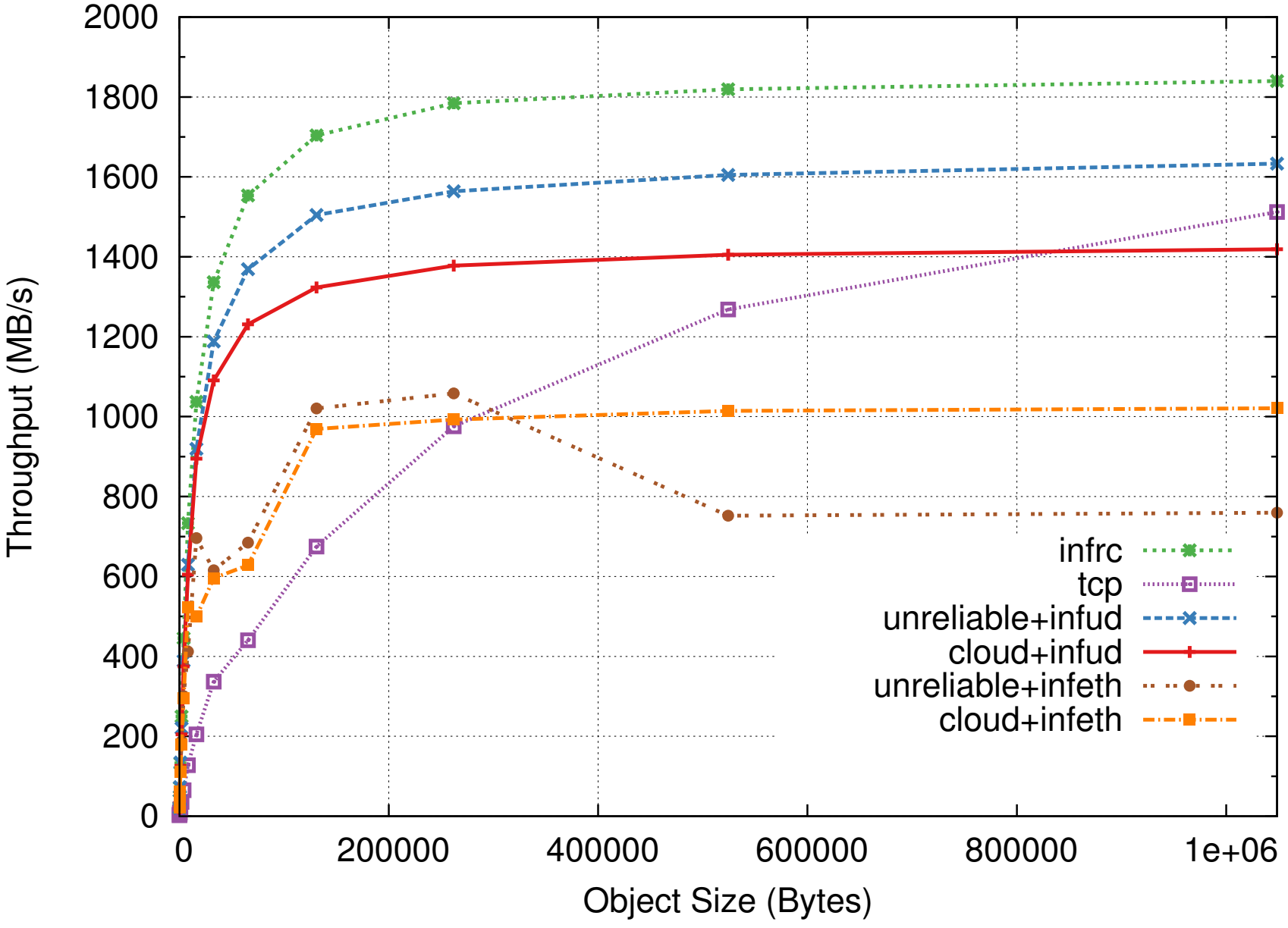
Preliminary Results

- One client and one storage server
- Load 1 GB of random data into server
- For each of various object sizes:
  Read random objects back-to-back
- Small object sizes show RPC latency
- Large object sizes show network utilization

RPC Latency


cloud+infeth
unreliable+infeth
cloud+infud
unreliable+infud
infrc
10 Gbps wire time
RTT (us)
Object Size (Bytes)
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
500
1000
1500
2000
2500
3000
3500
4000

Network Throughput



Weaknesses (Future Work)

- Don't understand why TCP's kernel crossings are so slow
- Don't understand our Ethernet driver's variance
- FastTransport performance isn't quite there yet
- Can't predict behavior in larger networks
  - Benchmark in slightly larger networks
  - Simulate datacenter networks

Conclusions

- We think we'll need a new transport protocol
- We're building a platform to experiment with different transports
- FastTransport is usable in small clusters
- Future work will expand to larger networks