Supporting Scalability & Adaptability via ADAptive Middleware & Network Transports (ADAMANT)

Joe Hoffert, Doug Schmidt

Vanderbilt University

Mahesh Balakrishnan, Ken Birman

Cornell University

Focus: Data Conferencing Applications

Characteristics

- Multiple continuous data streams
- Multiple senders/receivers
- Stream coordination, synchronization

Examples

- Search & rescue, targeting
- Stock update correlation
- Medical telemetry (e.g., wireless ER)
- Networked scientific application (e.g., weather monitoring)

Data Conferencing Challenges

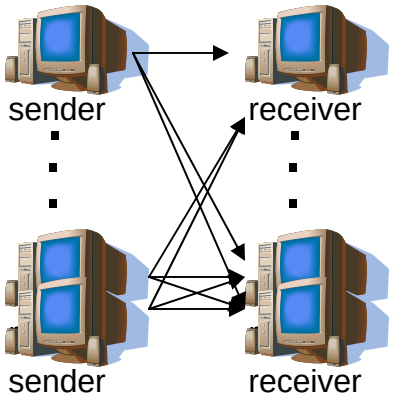
Incorporation of standards

- Pub/sub
- Real-time, QoS
- Heterogeneity



Scalability

- Number of receivers/senders
- Amount of data



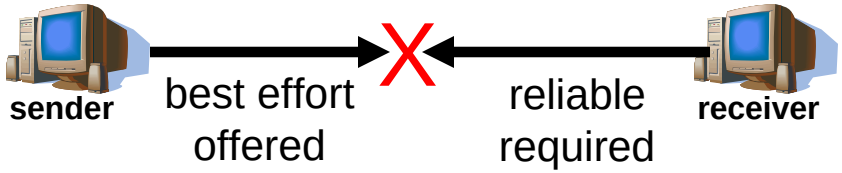
fast

&

reliable

Adaptability

- Contentious application requirements
- Variability in environment




```
Deadline_period = 'a'
```

QoS Configuration Complexity

- Semantic compatibility
- Accidental complexity

Solution Approach: ADAptive Middleware & Network Transports (ADAMANT)

Incorporation of standards

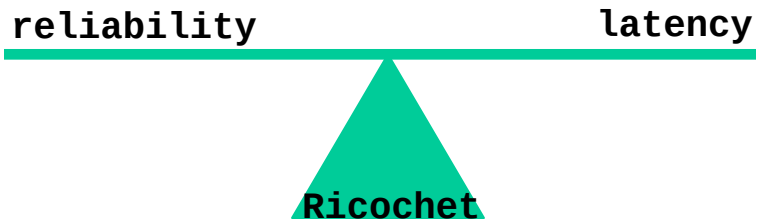
- Data Distribution Service (DDS) = OMG pub/sub standard
- 22 QoS policies
- Platform-agnostic IDL

OBJECT MANAGEMENT GROUP

DDS

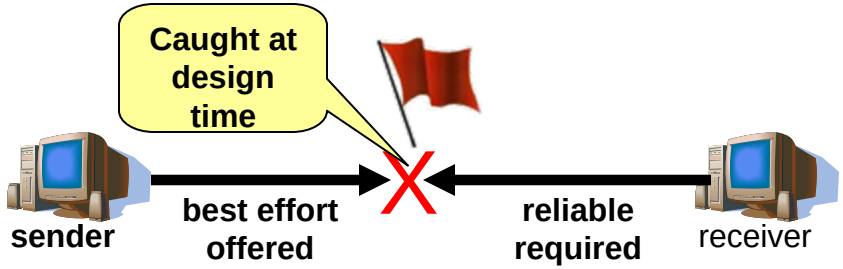
Scalability

- Data Distribution Service:
  - decouples senders, receivers
  - Implementations utilize multicast



Adaptability

- Ricochet++ transport framework provides composable modules
- Ricochet transport balances reliability & latency



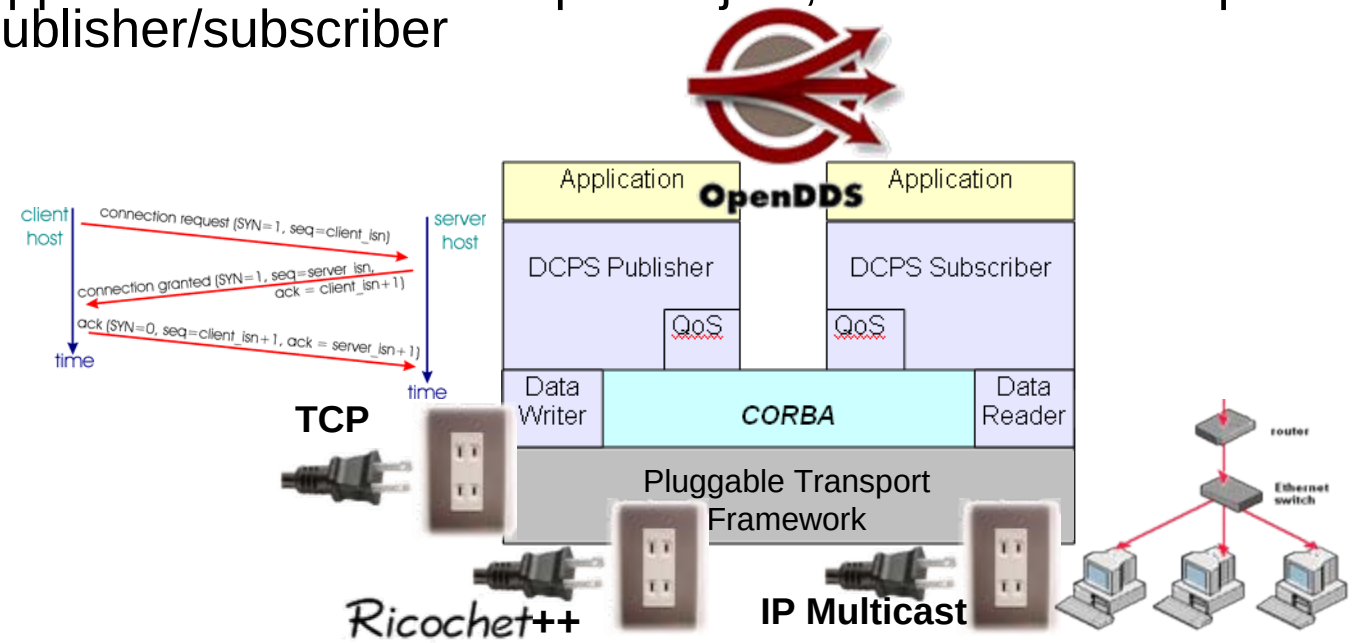
QoS Configuration Complexity

- DDS QoS Modeling Language:
  - Manages semantic compatibility
  - Reduces accidental complexity

ADAMANT Prototype

OpenDDS Pluggable Transport Framework supports:

- Standard transport protocols (e.g., TCP, UDP, IP multicast)
- Custom transport protocols
  - Inherit from key classes
  - Define custom behavior
  - Application creates transport object, associates transport with publisher/subscriber



Developed Ricochet++ pluggable transport

However, OpenDDS has limited DDS QoS and scalability support

ADAMANT Test Environment

Using Emulab Environment (www.emulab.net)

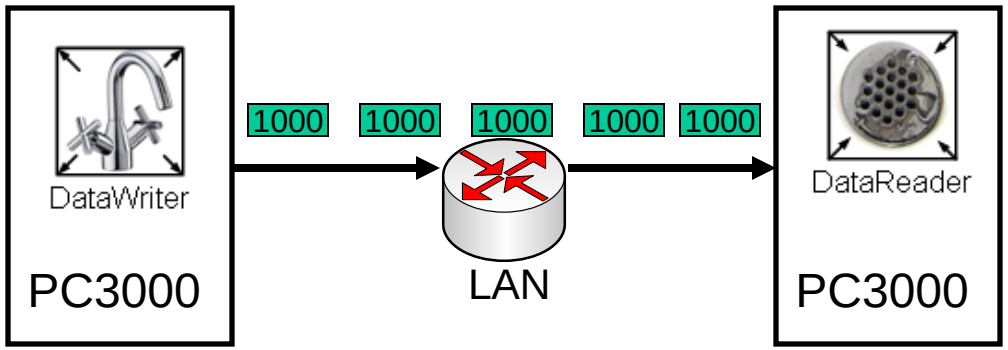
- PC3000 (64-bit Xeon, 3 GHz)
- Fedora Core 6
- Lossless LAN

Using DDSBench for benchmarking

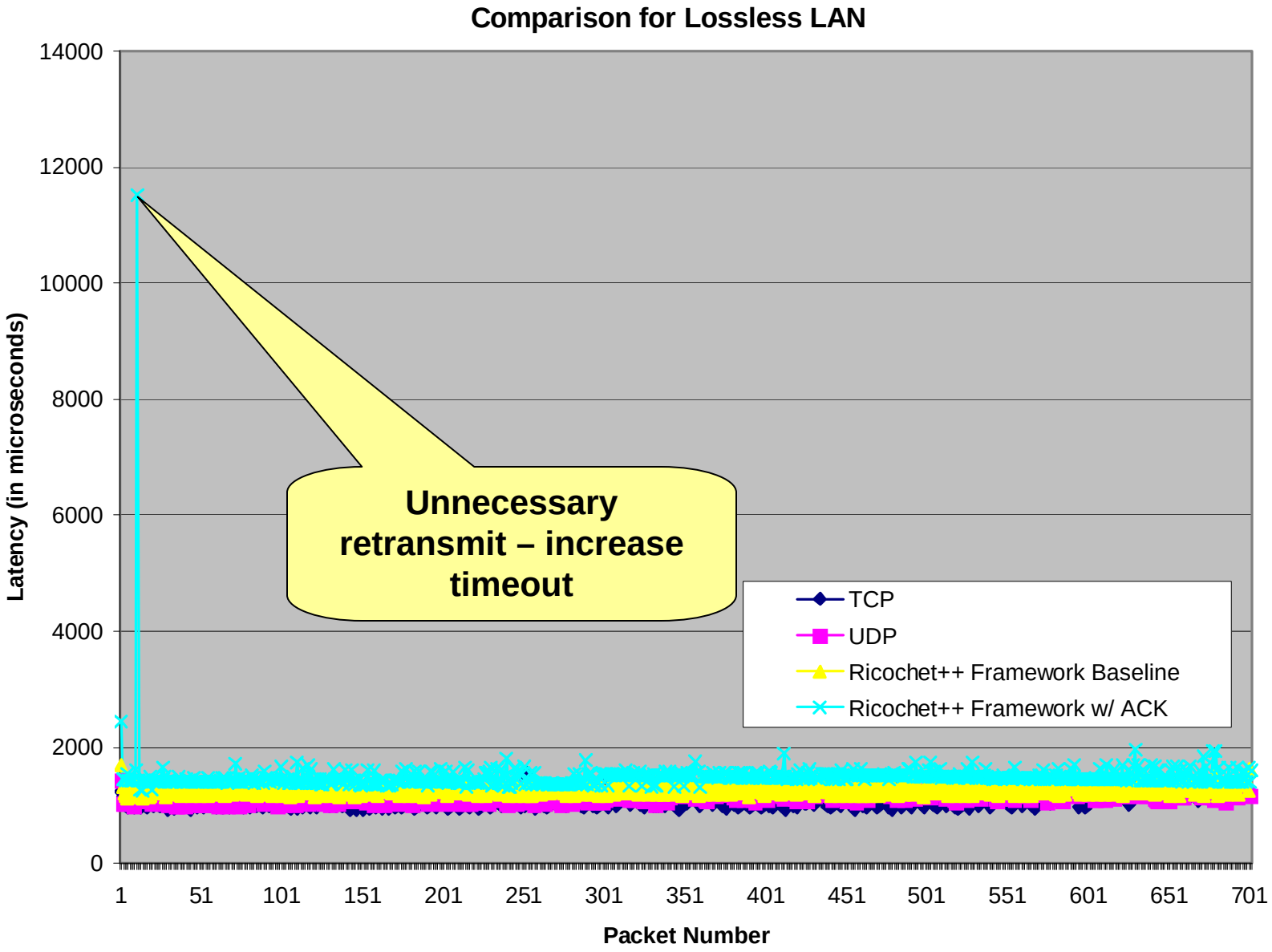
- Developed by MobiLab Research Group, Universita degli Studi di Napoli Federico II, Naples, Italy (http://www.mobilab.unina.it)
- Flexible interface for testing DDS implementations

Initial testing scenario

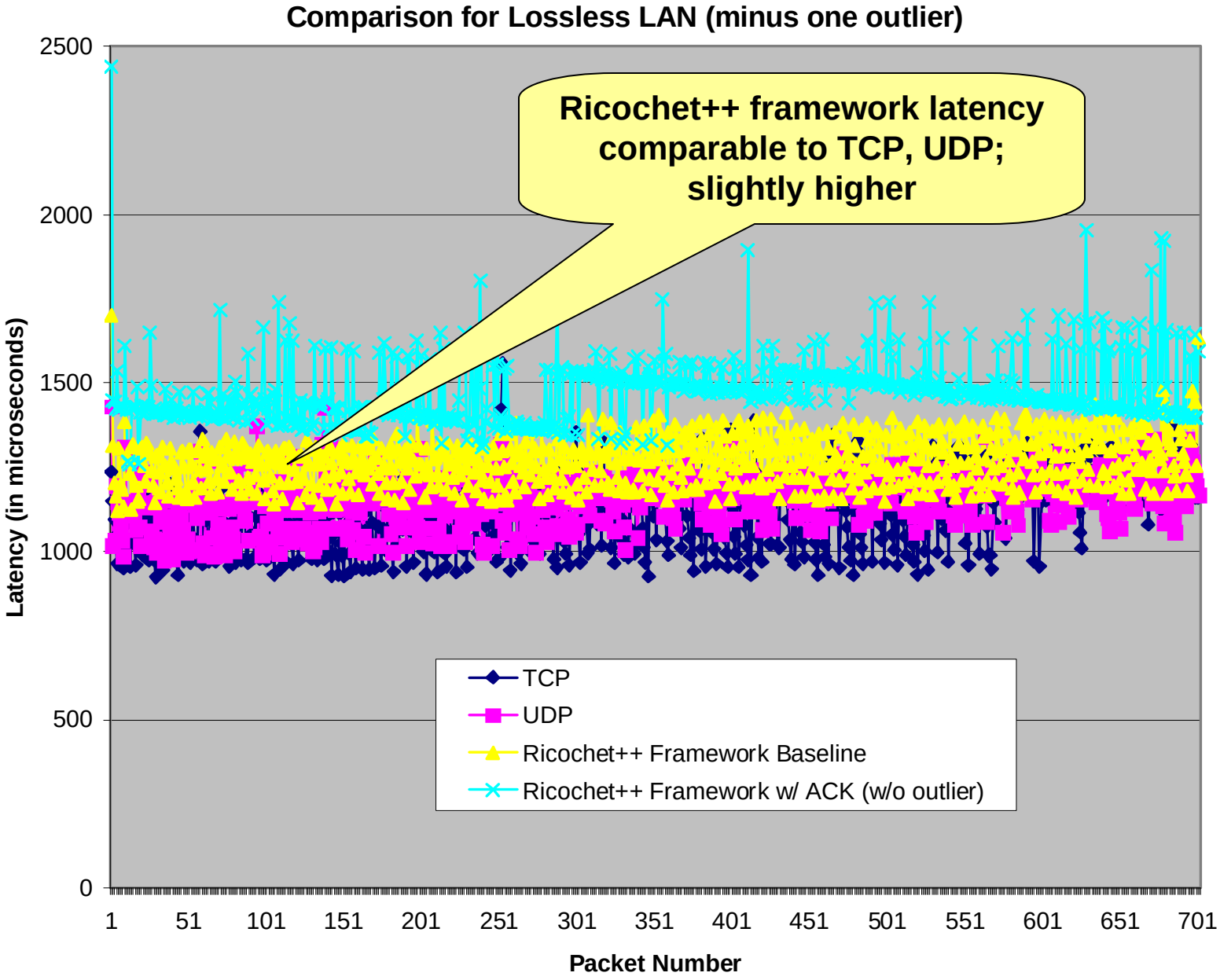
- Using latest OpenDDS
- 1 data writer on 1 machine, 1 data reader on another machine
- Data packet size of 1,000 bytes
- 700 messages sent



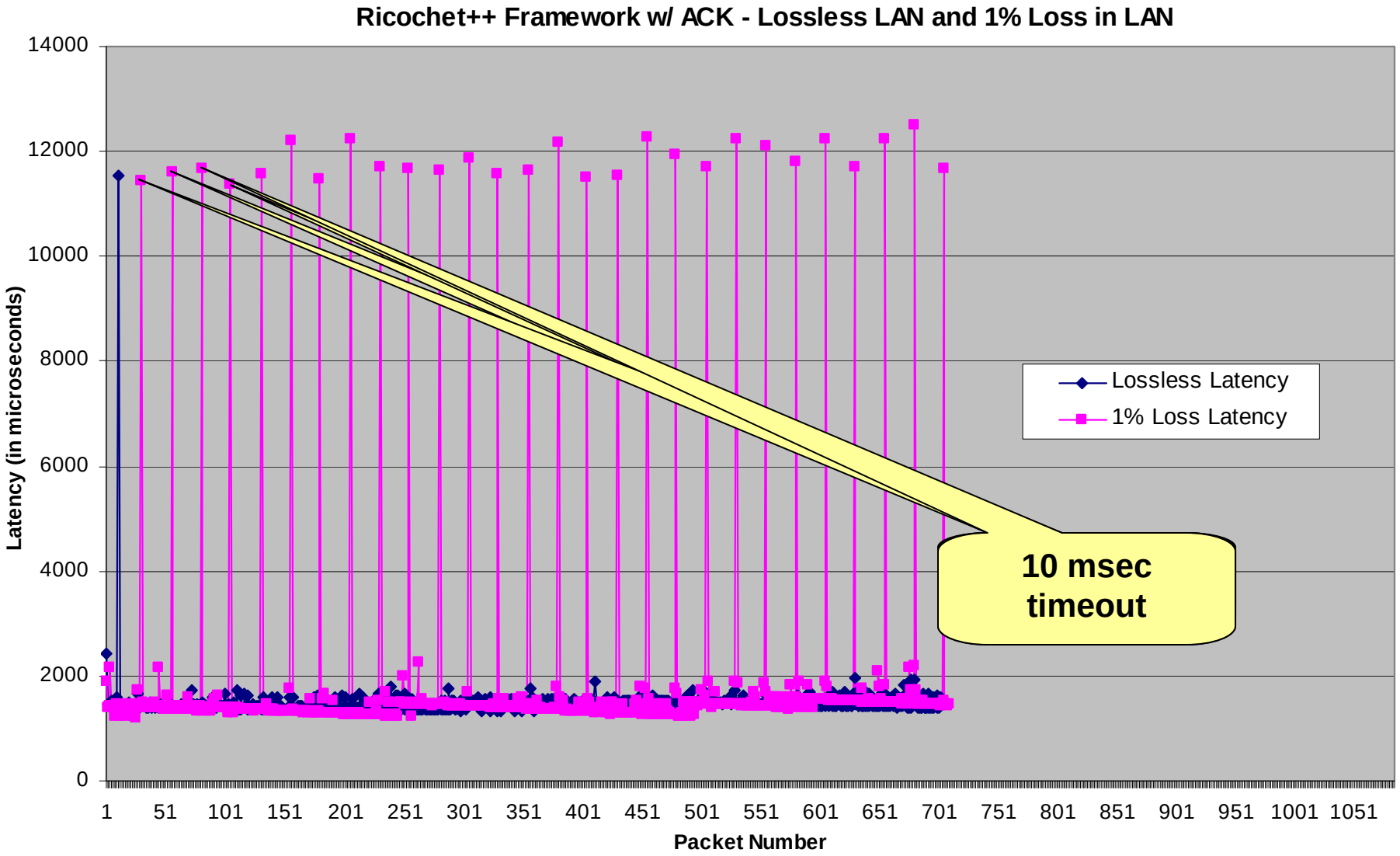
Empirical Results



Empirical Results



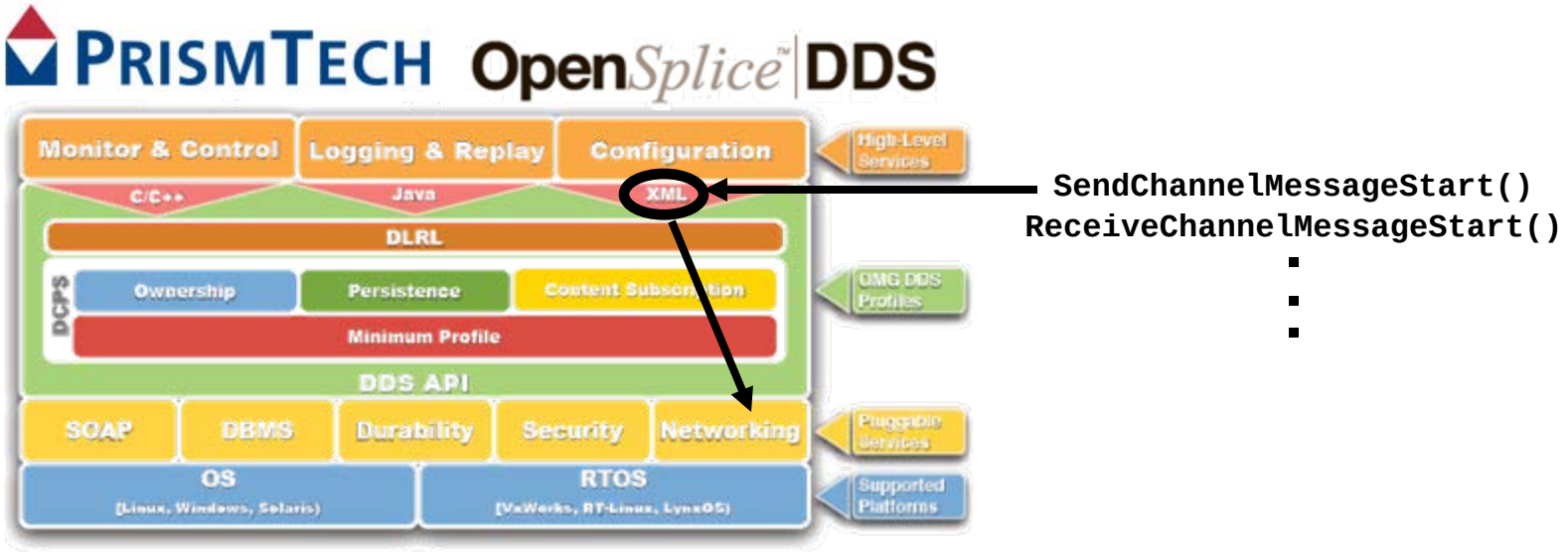
Empirical Results



Next Steps: Enhanced ADAMANT Prototype

PrismTech's OpenSplice DDS network plug-in provides:

- Efficient C API
- XML configuration file to specify transport functions used

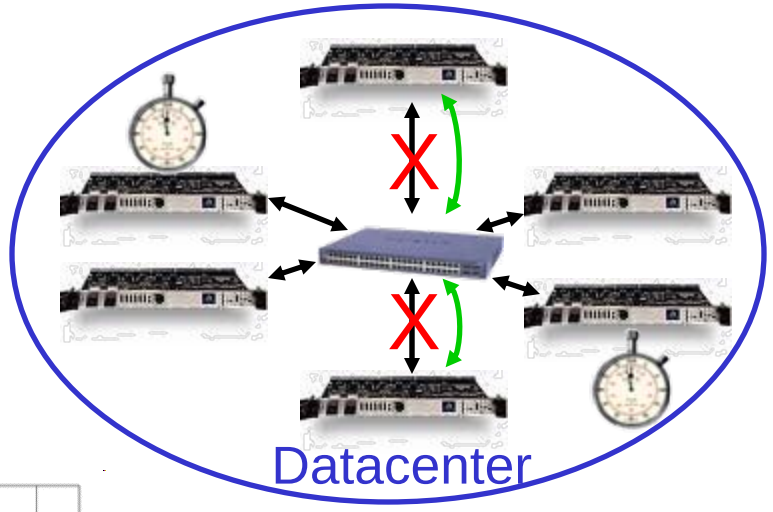


Developing Ricochet++ plug-in to leverage more extensive and scalable DDS support than OpenDDS

Next Steps: Evaluate Data Distribution Profiles

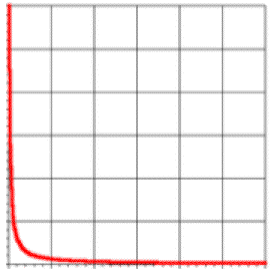
Effect on transport protocol behavior?

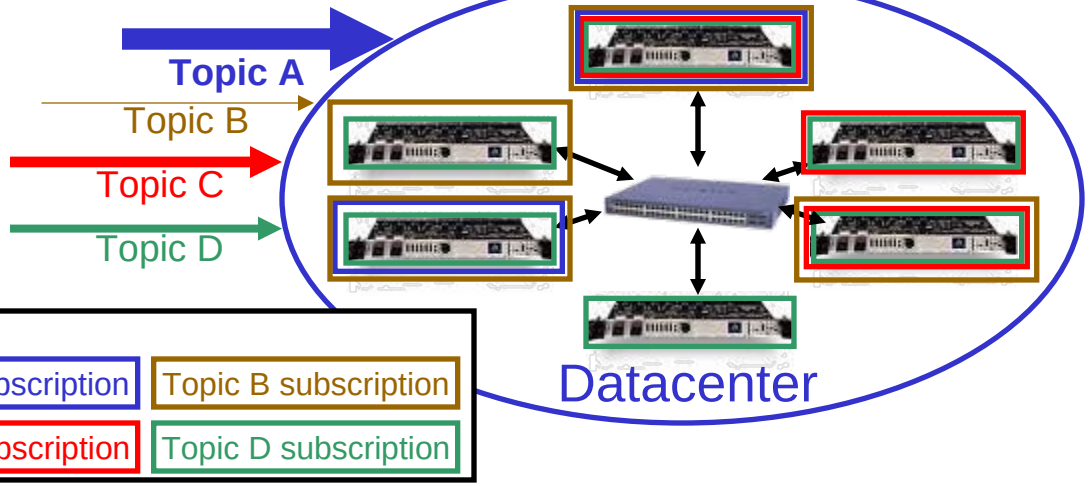
- Recovery of data for reliability
- Latency of recovery
- Data "staleness" (i.e., state of system as it should be & as it is)



Data distribution profiles

- Characterization, e.g.,
  - Zipf
  - Uniform
- Transmission distribution (e.g., most data is for one topic, topic A)
- Subscription distribution (e.g., one topic has more subscribers than any other, topic D)
- Taxonomy: relevance, importance

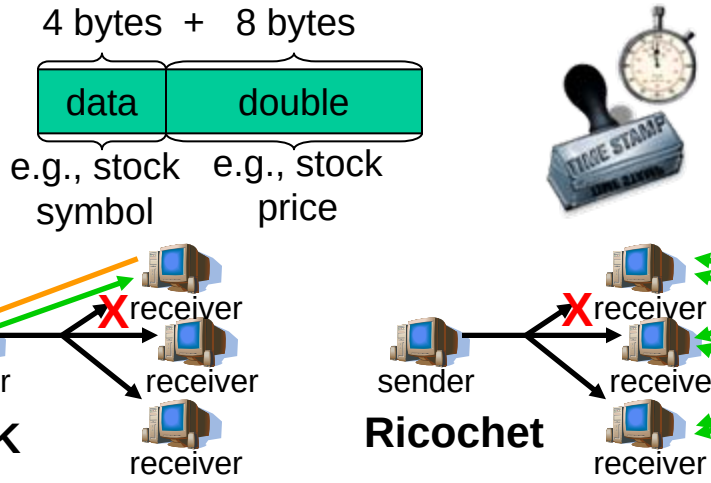




Data Distribution Profiling Harness

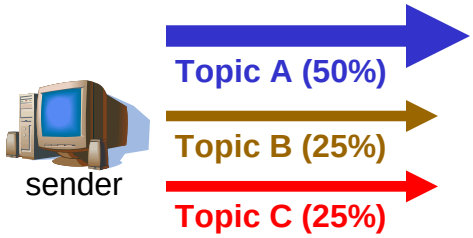
Initial implementation developed

- Fixed message size (12 bytes)
- High resolution timestamps (microseconds)
- Variable Ricochet++ transport protocol (e.g., NAK, Ricochet)



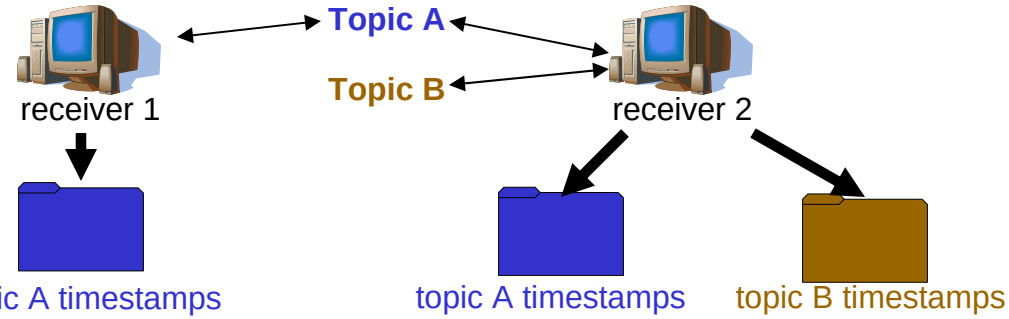
One Sender

- Specify topic distribution percentages (transmission distribution), e.g., 50% topic A, 25% topic B, 25% topic C



Multiple receivers

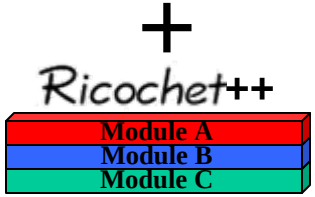
- Specify topics to join (subscription distribution)
- Received message timestamps written to files



Concluding Remarks

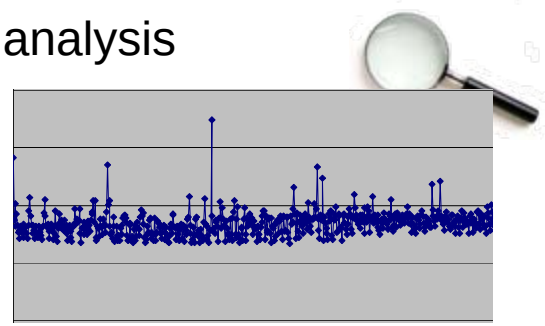
ADAMANT[1] progress to date

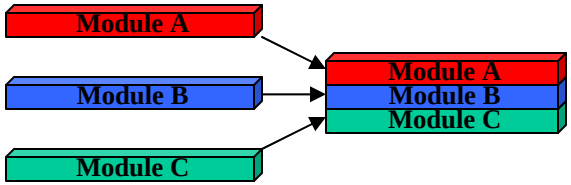
- Ricochet++[2] framework in place
- OpenDDS[3], OpenSplice[4] support
- DQML developed in GME

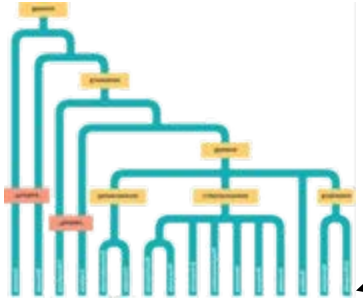


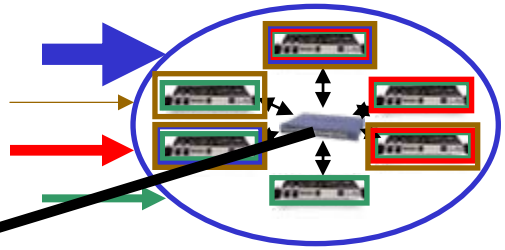
Work to do

- In-depth performance testing & analysis
  - For individual modules
  - For groups of modules
  - vs. standard protocols
- Better Ricochet++ integration needed with OpenSplice, OpenDDS
- Characterize distribution profiles
- Develop DQML interpreters for OpenDDS, OpenSplice









[1]http://www.dre.vanderbilt.edu/~jhoffert/ADAMANT
[2]http://www.cs.cornell.edu/projects/quicksilver/Ricochet.html
[3]http://www.opendds.org
[4]http://www.prismtechnologies.com/