



Workshop on Distributed Object Computing
for Real-time and Embedded Systems

July 14 - 16, 2008, Washington, DC, USA



Supporting Scalability and Adaptability via ADaptive Middleware And Network Transports (ADAMANT)

Joe Hoffert, Doug Schmidt

Vanderbilt University

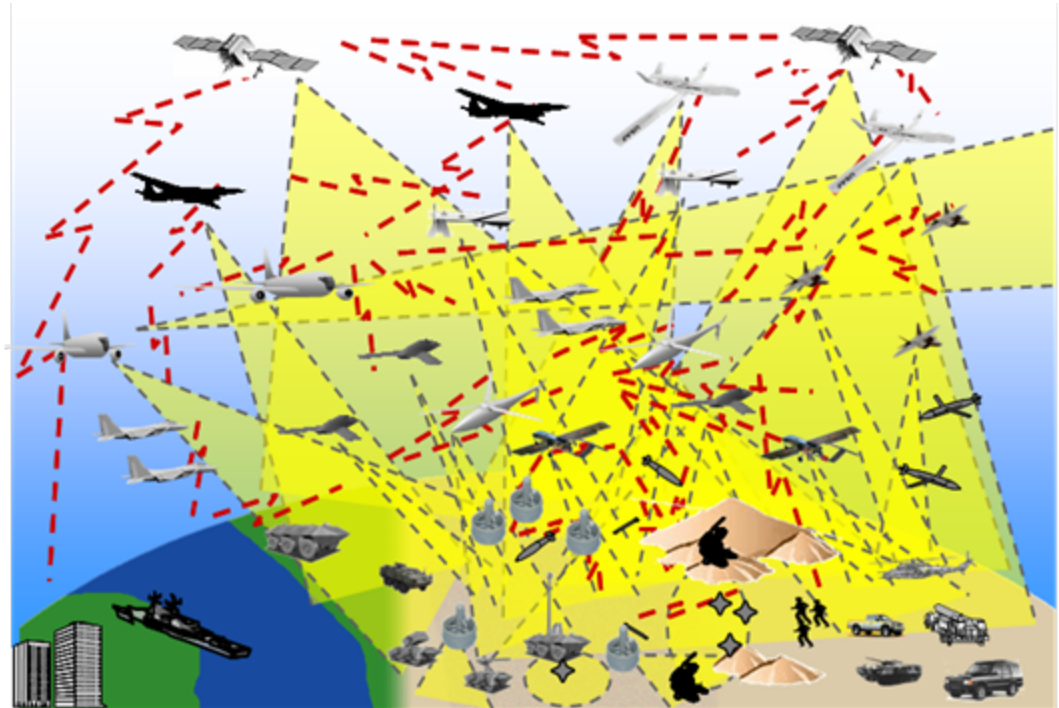
Mahesh Balakrishnan, Ken Birman

Cornell University



• Motivation

- Existing SOA standards & technologies in use for large-scale data-centric platforms do not focus on timely response
- Military & emergency systems very often need rapid response, scalability, bandwidth guarantees, fault-tolerance
- ... & need to function under stressful conditions, over connections with “uncivilized” properties, e.g., bursty loss, latency issues, route flaps

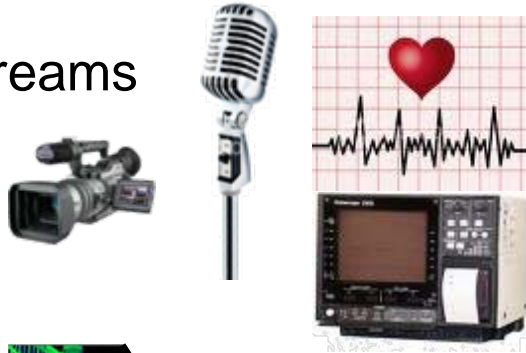




Data Conferencing Applications

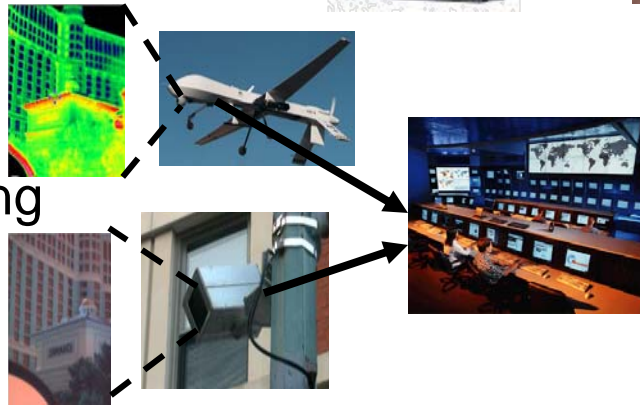
• Characteristics

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



• Examples

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



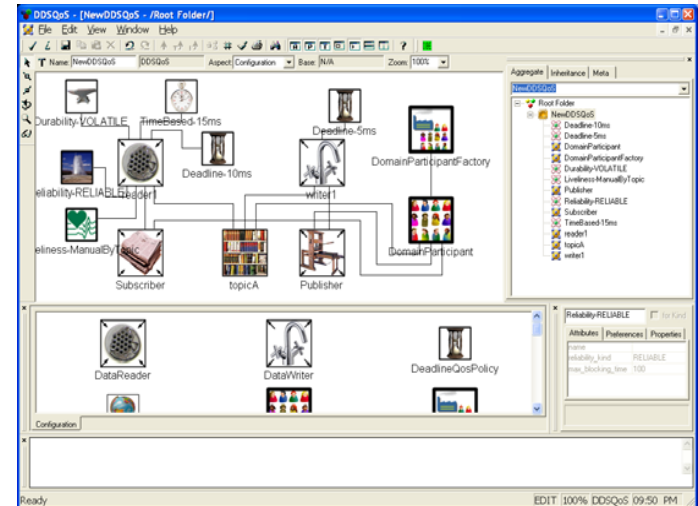


Solution Approach: ADaptive Middleware And Network Transports (ADAMANT)



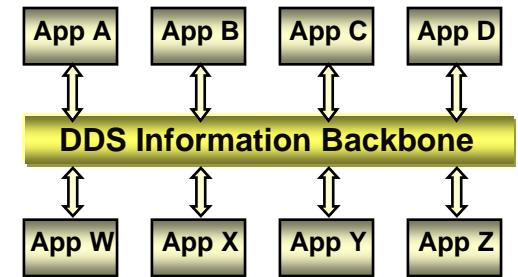
DDS QoS Modeling Language (DQML)

- Domain-specific modeling language
- Abstract away low-level tedious implementation details
- Manage QoS semantic compatibility
- Encapsulate higher-level QoS profiles for application types



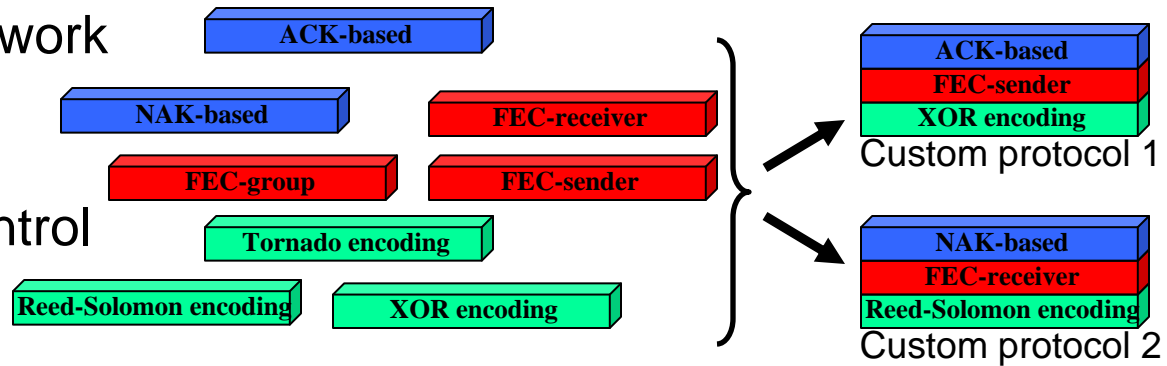
Data Distribution Service

- Robust standardized pub/sub API
- Fine-grained QoS policy control



Ricochet++

- Transport protocol framework
- Composable modules
- Autonomic adaptation
- Fine-grained protocol control



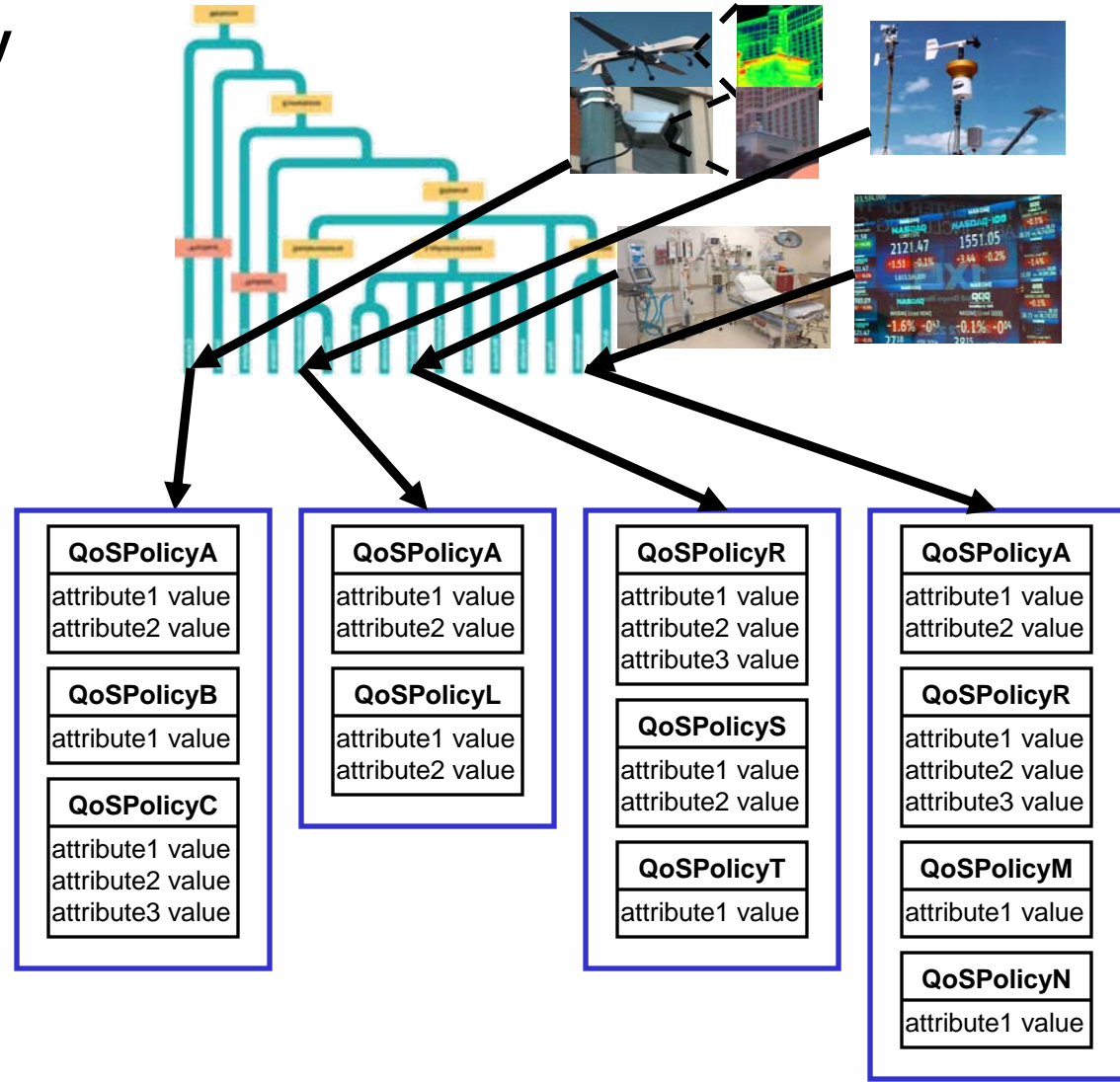


Developing Data Conferencing Taxonomies & Configuration Patterns



Data conferencing variability

- Reliability
- Deadlines
- Data type
- Amount of data
- Frequency of data updates



DQML to leverage application taxonomies

- Map to DDS QoS policies
- Use/Create DDS policy patterns



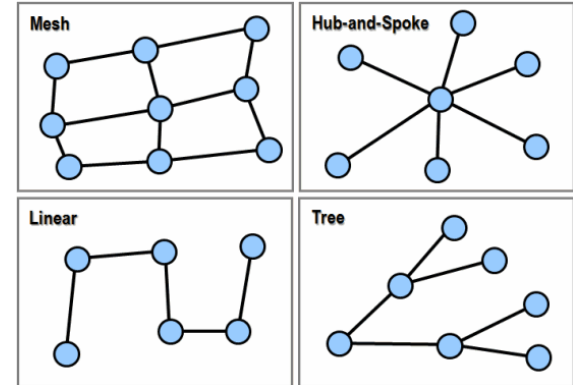
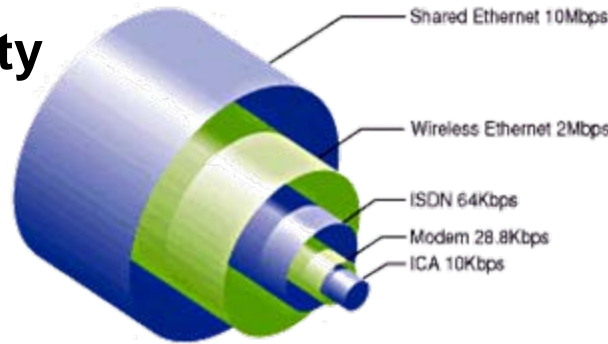


Maintaining Specified QoS via Ricochet++ Autonomic Adaptation



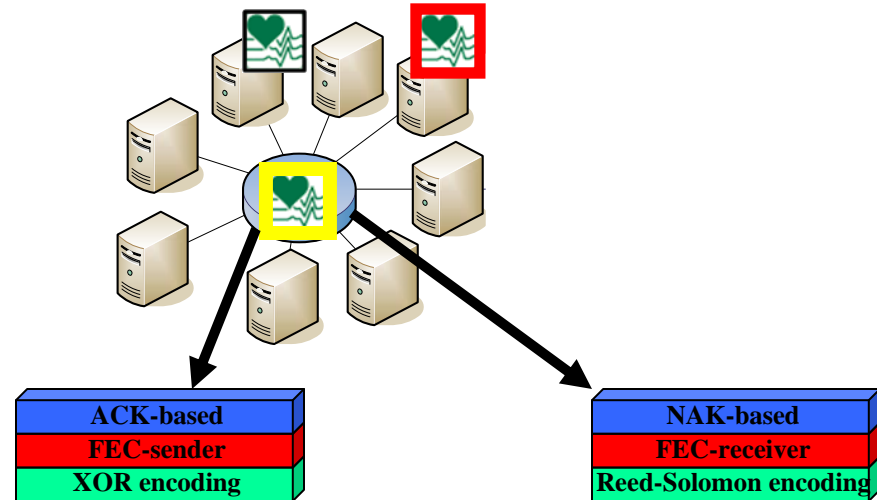
Environment Variability

- Network bandwidth
- Latency
- Network congestion
- Network topology
- Route flaps



Ricochet++ for adaptation

- Environment monitored for relevant changes
- Autonomic adaptation based on environment
- Maintain required QoS



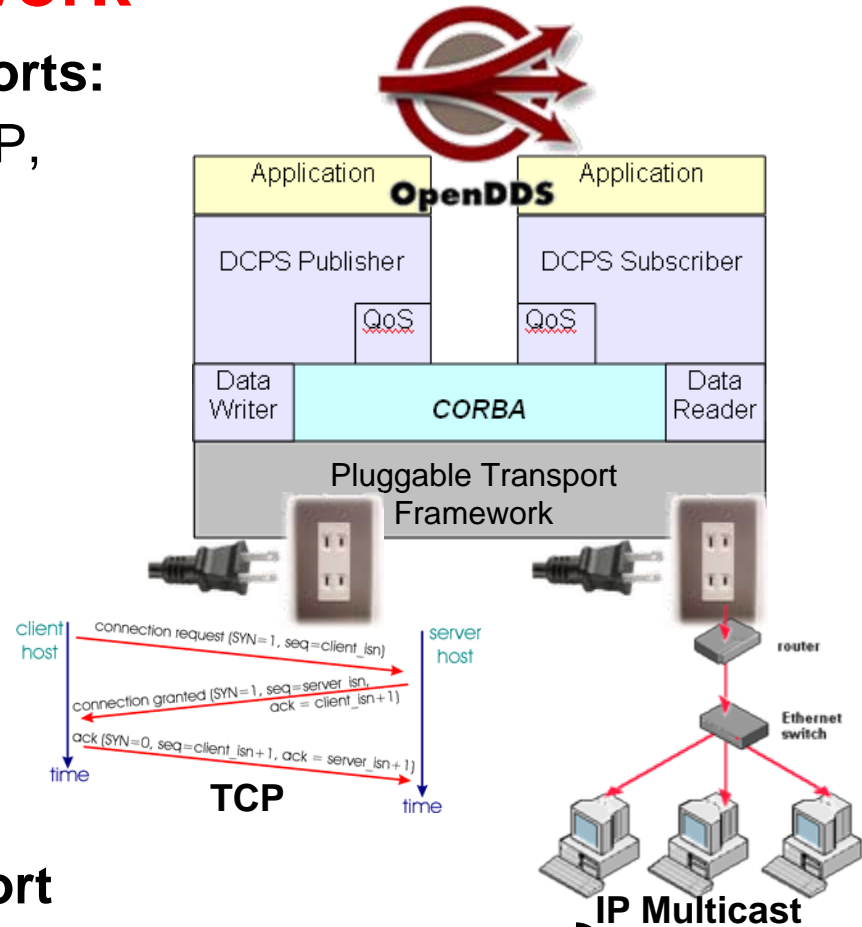
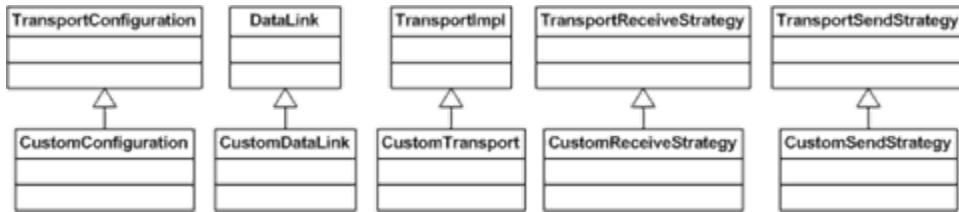


OpenDDS Pluggable Transport Framework



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

```
// Create the Ricochet transport and configuration objects.
OpenDDS::DCPS::TransportImpl_rch ricochet_impl
= TheTransportFactory->create_transport_impl (transport_impl_id, "Ricochet", OpenDDS::DCPS::DONT_AUTO_CONFIG);
OpenDDS::DCPS::TransportConfiguration_rch config
= TheTransportFactory->create_configuration (transport_impl_id, "Ricochet");
⋮
// Attach the transport to the publisher.
OpenDDS::DCPS::AttachStatus status = publisher_impl->attach_transport(ricochet_impl.in());
```





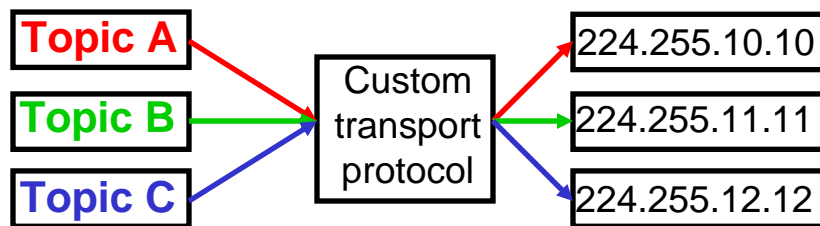
OpenDDS Pluggable Transport Framework: R&D Challenges



Application developer must manage correlation between:

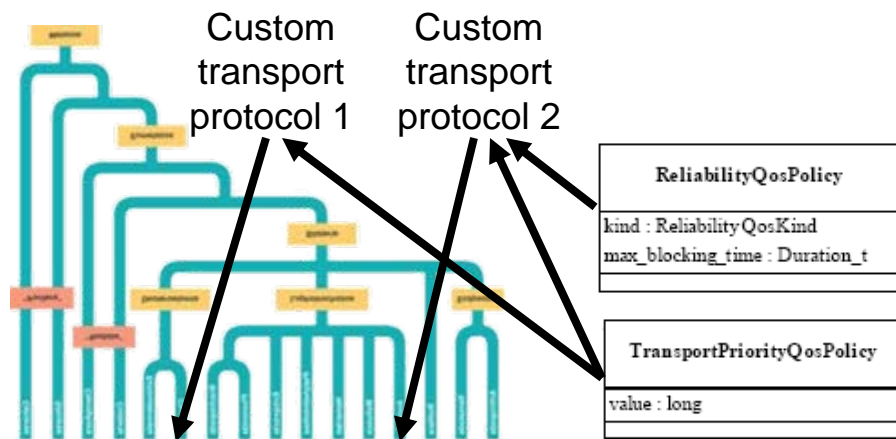
- QoS policies and transport protocols
- Transport protocols and topics
 - Transport protocols associated with publisher/subscriber
 - Topic associated with data reader/writer

```
// Create the UDP transport protocol.
OpenDDS::DCPS::TransportImpl_rch udp_impl
  = TheTransportFactory->create_transport_impl (transport_impl_id, "SimpleUdp",
                                               OpenDDS::DCPS::DONT_AUTO_CONFIG);
:
:
// Attach the UDP transport to the publisher.
OpenDDS::DCPS::AttachStatus status = pub_impl->attach_transport(udp_impl.in());
:
:
// Create the datawriter and add reliability QoS.
DDS::DataWriterQos dw_qos;
pub->get_default_datawriter_qos (dw_qos);
dw_qos.reliability.kind = ::DDS::RELIABLE_RELIABILITY_QOS;
DDS::DataWriter_var dw =
pub->create_datawriter(topic.in (), dw_qos, DDS::DataWriterListener::_nil());
```



Develop taxonomy of transport protocols

- Protocols registered based on properties in taxonomy
- Middleware maps relevant QoS policies to appropriate transport protocol(s)



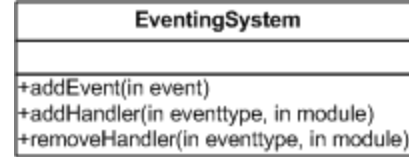


Ricochet++ Transport Protocol Framework: Status



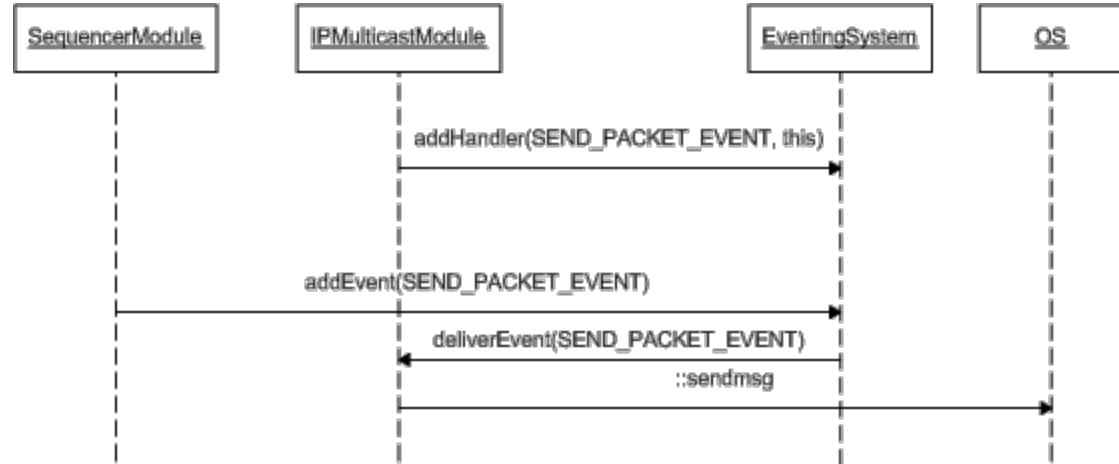
Eventing architecture

- EventingSystem singleton per process
- Defines events of interest, e.g.,
 - SEND_PACKET_EVENT
 - GOT_PACKET_EVENT
- Provides event injection into the system
- Provides event registration for modules



Protocol Modules

- Register for events of interest
- Define behavior when event is delivered
- Inject events of interest
- Existing modules
 - IPMulticastModule
 - SequencerModule
 - MuxModule
 - AckModule
 - NakModule



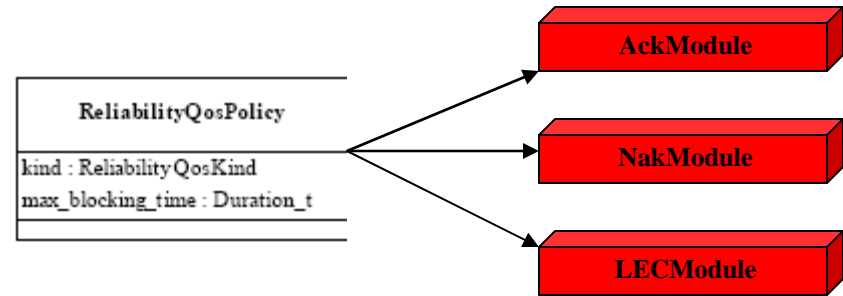


Ricochet++ Transport Protocol Framework: R&D Challenges



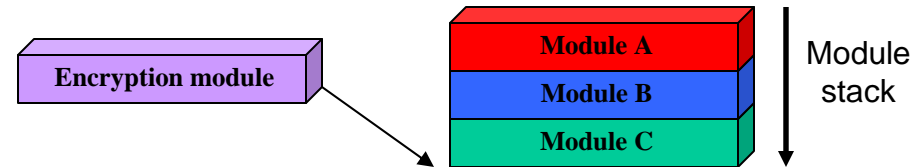
Mapping QoS policy values to Ricochet module properties

- Reliability -> ACK or NAK? both?
- Lateral Error Correction (LEC) between reliable and best-effort
- Have DSML handle this (i.e., leapfrog DDS QoS policies)?



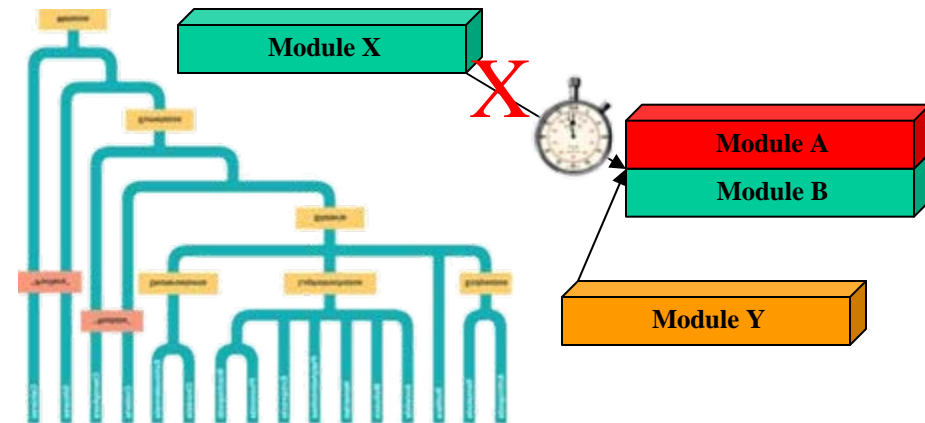
Managing module flexibility

- Module ordering, e.g.,
 - encryption after data manipulation
 - previous module consumes data needed by next module
- mutual exclusion (e.g., multiple encryption policies)
- timeliness, symmetry



Automated management for modules

- develop taxonomy of policy types
- insertion rules (e.g., if/conflict resolution, where)
- run-time support





Preliminary Test Environment

Using Emulab Environment

(www.emulab.net)

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



Using DDSBench for benchmarking

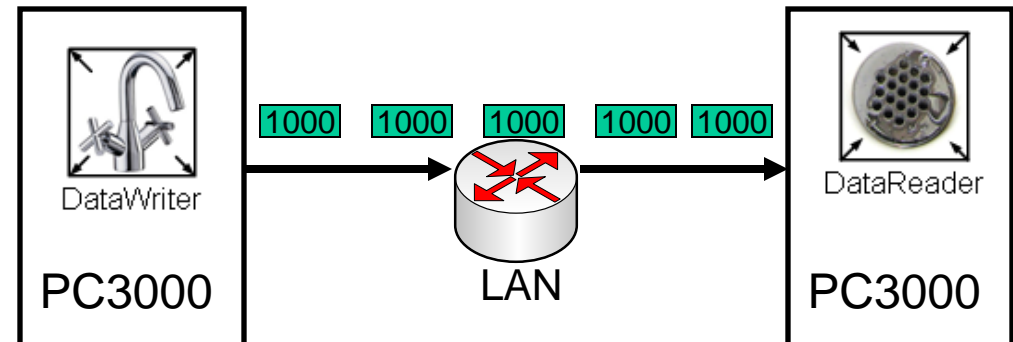
- Developed by MobiLab Research Group, Universita degli Studi di Napoli Federico II, Naples, Italy (<http://www.mobilab.unina.it>)
- Flexible interface for testing DDS implementations



OpenSplice™ | DDS

Testing scenario

- Using latest OpenDDS
- 1 data writer on one machine, 1 data reader on another machine
- Data packet size of 1000 bytes
- 700 messages sent

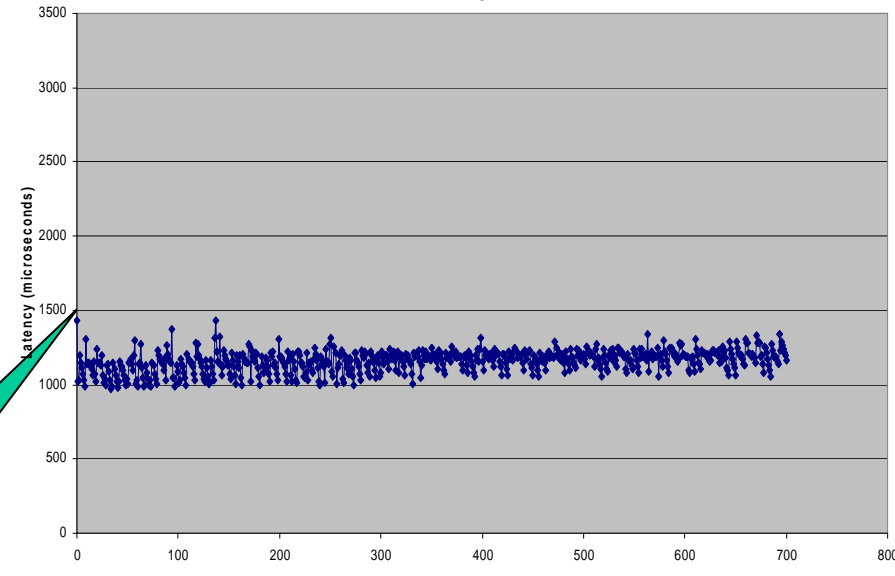
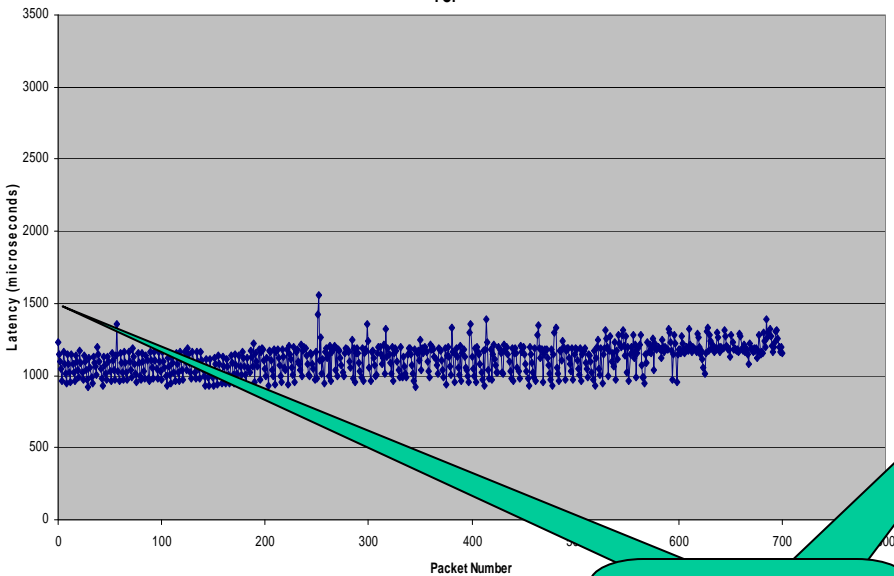




Preliminary Results

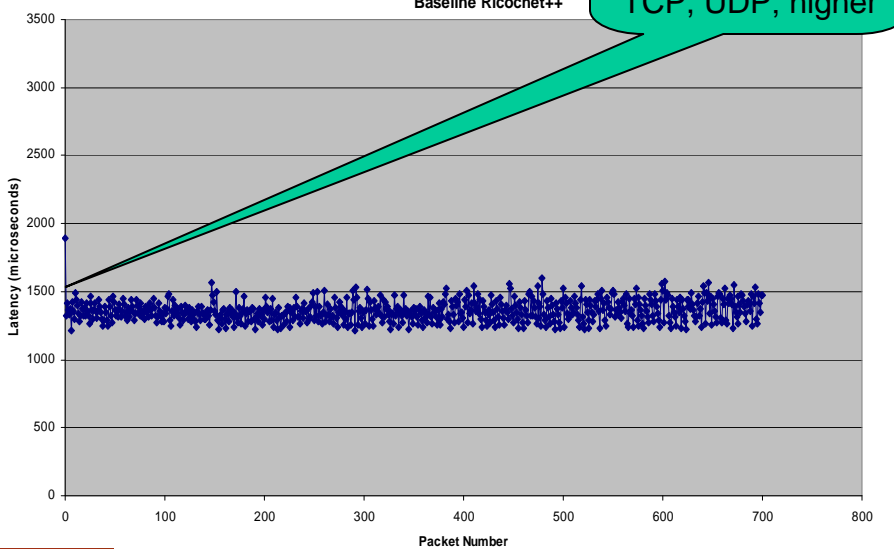
TCP

UDP

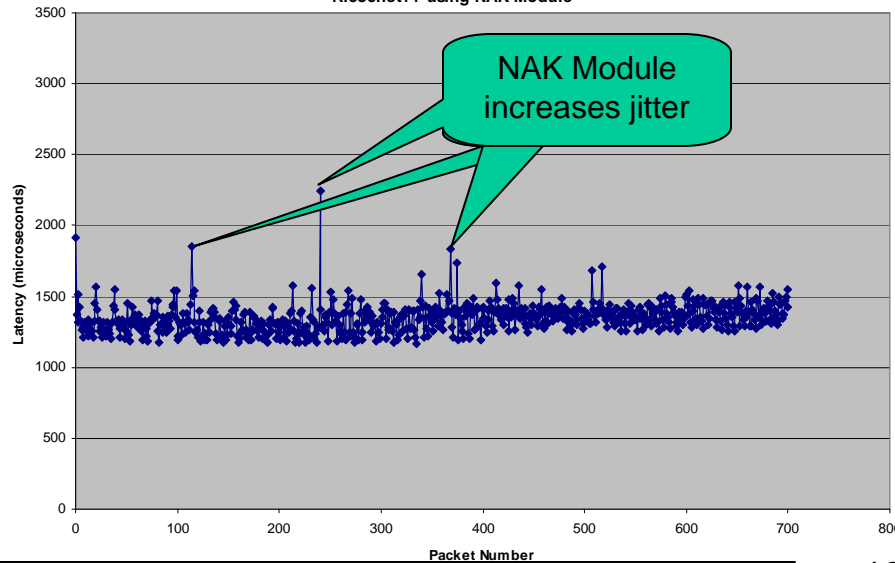


Ricochet latency comparable to TCP, UDP; higher

Baseline Ricochet++



Ricochet++ using NAK Module



NAK Module increases jitter





Concluding Remarks

Initial ADAMANT¹ work started

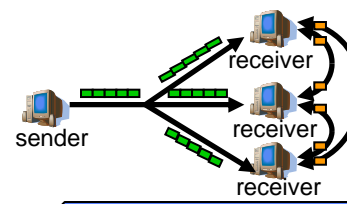
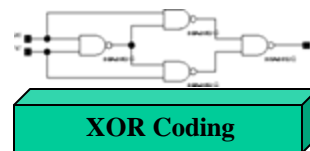
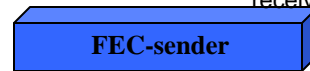
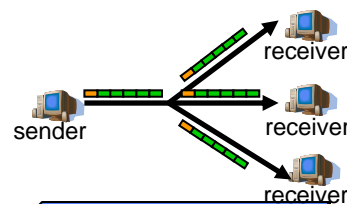
- Ricochet++² framework in place
- Integrated with OpenDDS³
- Composable transport modules promising research area
- Flexible
- More modules needed



OpenDDS

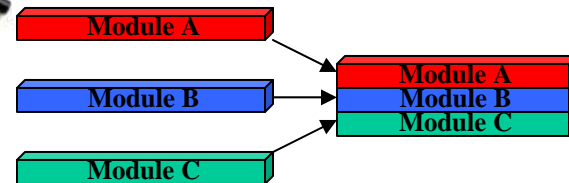
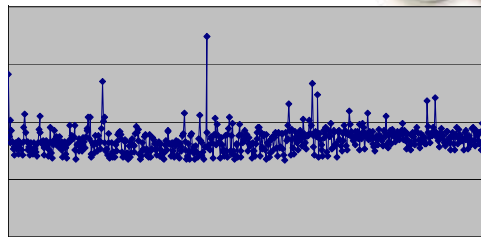


Ricochet++



Lessons learned

- More in-depth performance analysis needed
 - For individual modules
 - For groups of modules
- More seamless integration with Ricochet++ and OpenDDS
- Fine tune modules
 - Reduce baseline latency
 - Reduce jitter



¹<http://www.dre.vanderbilt.edu/~jhoffert/ADAMANT>

²<http://www.cs.cornell.edu/projects/quicksilver/Ricochet.html>

³<http://www.opendds.org>