Fault-Tolerant Reliable Delivery of Messages in Distributed Publish/Subscribe Systems

Shrideep Pallickara, Hasan Bulut & Geoffrey Fox
Community Grids Lab
Indiana University
NaradaBrokering

• Content distribution infrastructure based on the publish/subscribe paradigm
  ▪ Support for variety of QoS for the streams
  ▪ Information assurance

• Open-source project http://www.naradabrokering.org

• Deployed in a wide variety of domains
  ▪ GIS, Audio/Video conferencing, collaboration, Geoscience and High Energy Physics

• Current release 2.0.2
  ▪ 1425 classes, 157 packages and 300,000 lines of code
Reliable Delivery: Desiderata

• Cope with node/link failures & unpredictable links
  ▪ Links may duplicate, garble and lose messages

• Many-to-Many reliable delivery across sessions
  ▪ Support recovery from failures or disconnects
  ▪ Support replays

• Transport-independent

• Support exactly-once delivery
  ▪ Order, duplicate detection

• Authorized reliable delivery
Reliable Delivery

- Persist Events
- Maintain Registered Entities
- Track Delivery Sequences

- One Repository associated with a reliable topic
- A Repository can manage multiple reliable topics
**Reliable Delivery**

- **PUBLISHER**
  - P2R-Order
  - R2P-NAK
  - P2R-Re transmit
  - R2P-ACK

- **REPOSITORY**
  - R2S*-Persistent
  - S2R-ACK
  - R2S-Rectify
  - S2R-NAK
  - R2S-Re transmit
  - R2S-Sync

- **SUBSCRIBER**

**Lines Indication:**
- **Black** Regular Exchange
- **Red** Error Detection
- **Blue** Error Correction
Reliable delivery: Advantages

- Reliable delivery ONLY for **authorized** entities
  - Coexists with entities not interested in reliable delivery
- Storage, is **not communal**, and should be provisioned by the topic owner.
- Control Messages are issued over different topics.
  - Discovery constraints can be imposed e.g. Restrict replays
- Different **QoS** can be associated with control topics.
  - Require signed acknowledgements (Non-repudiation)
  - Buffering & Jitter reduction services for replayed messages.
    - Streams replayed at say 24 fps instead of 500 fps
- Easy to maintain **audit trails**
  - Track client loss rates, NAKs, disconnects & recoveries
- Lends itself naturally for greater redundancy
Repository Redundancy

\{R_{A1}, R_{A2}, R_{A3}, R_{A4}\} \rightarrow \text{Repository bundle for reliable topic A}
Repository Redundancy

- Multiple repositories constitute a repository bundle
  - A given repository can be part of multiple bundles
- Associate a repository-bundle for a given topic for greater redundancy
  - Sustain repository failures and downtimes
- **Fine tune** redundancy associated with a bundle
  - Graceful addition & removal of constituent repositories
- Clients leverage network **proximity** through bundle
  - By choosing a *closer* repository, communication latencies are reduced. Retransmissions and recoveries are faster.
- Repository with which a client actively interacts with is its **steering repository**.
- Set aside repositories for recovery and replays
Delivery overheads in different Topologies for different message payload sizes

- 3 Brokers, Best effort
- 3 Brokers, 1 Repository
- 3 Brokers, 3 Repositories
Ongoing Activities

• Repository placement schemes
  ▪ Reduction of publisher & subscriber overheads
  ▪ Facilitate faster recoveries and error-corrections
  ▪ Dedicated repositories for use in replays

• Current Deployment
  ▪ eSports System – To facilitate recording, annotation and replays of multimedia streams

• Release Schedule
  ▪ Will be released as part of NaradaBrokering 3.0 in June 2007