Grid and Internet Messaging Systems

EuroPVM/MPI’03 Venice 30 September 2003

PTLIU Laboratory for Community Grids
Geoffrey Fox
Computer Science, Informatics, Physics
Indiana University, Bloomington IN 47404

http://grids.ucs.indiana.edu/ptliupages
Point of Talk

- **PVM, MPI** and related systems were developed to support the needed functionality and performance of message-based programming and execution models of parallel computing.

- **Peer-to-peer networks, Web Services** (and hence the Grid via OGSA) and **event-based collaboration** models and built around “distributed objects”, “processes”, “programs”, “services” etc. exchanging **messages**.

- What is an **appropriate messaging** model in this second case of **distributed systems**?

- We will illustrate the discussion with **NaradaBrokering** built by Shrideep Pallickara at **http://www.naradabrokering.org**.
What is a distributed memory system?

- We might wish to consider three classes of multi-node computers
- 1) **Classic MPP** with microsecond latency and scalable internode bandwidth ($t_{comm}/t_{calc} \sim 10$ or so)
- 2) **Classic Cluster** which can vary from configurations like 1) to 3) but typically have 0.1 millisecond latency and modest bandwidth
- 3) **Classic Grid** or distributed systems of computers around the network
  - Latencies of inter-node communication – 100’s of milliseconds but can have good bandwidth

- All can have same peak CPU performance (if same nodes) but time cost in synchronization increases as one goes from 1) to 3)
- **Cost of system** (dollars per gigaflop) can decrease by factors of 2 at each step from 1) to 2) to 3)
- One should typically **not use classic MPP** if class 2) or 3) suffices unless some security or data issues dominates over cost-performance
- One can **not** typically use a **Grid as a true parallel computer** – it can link parallel computers together for convenient access etc.
NaradaBrokering

NaradaBrokering Broker Network

Audio/Video Conferencing Client

Brokers ONLY process messages

Minicomputer
Firewall
Computer
Modem
Server
Laptop computer
PDA
Audio/Video Conferencing Client
Workstation
Peers
“GridMPI” v. NaradaBrokering

- In parallel computing, MPI and PVM provided “all the features one needed” for inter-node messaging
- NB aims to play same role for the Grid but the requirements and constraints are very different
  - NB is not MPI ported to a Grid/Globus environment
- Typically MPI aiming at microsecond latency but for Grid, time scales are different
  - 100 millisecond quite normal network latency
  - 30 millisecond typical packet time sensitivity (this is one audio or video frame) but even here can buffer 10-100 frames on client (conferencing to streaming)
  - <1 millisecond is time for a Java server to “think”
- Jitter in latency (transit time through broker) due to routing, processing (in NB) or packet loss recovery is important property
- Grids need and can use software supported message functions and trade-offs between hardware and software routing different from parallel computing
NaradaBrokering

- Based on a network of cooperating broker nodes
  - Cluster based architecture allows system to scale in size
  - Grid routing topologies are open research issue?

- Originally designed to provide uniform software multicast to support real-time collaboration linked to publish-subscribe for asynchronous systems.

- Perhaps better thought of as stream not message handler

- Now has several core functions
  - Reliable order-preserving “Optimized” Message transport (based on performance measurement) in heterogeneous multi-link fashion with TCP, UDP, SSL, HTTP, and will add GridFTP
  - General publish-subscribe including JMS & JXTA and support for RTP-based audio/video conferencing
  - General software routing to avoid network problem
  - Distributed XML event selection using XPATH metaphor
  - QoS, Security profiles for sent and received messages
  - Interface with reliable storage for persistent events
Laudable Features of NaradaBrokering

- Is open source [http://www.naradabrokering.org](http://www.naradabrokering.org) available now; major new release for SC03
- Will have end-point “plug-in” as well as standalone brokers
  - end-point is service or user-interface machine
- Will have a discovery service to find nearest brokers
- Does tunnel through most firewalls without requiring ports to be opened
- Links to NWS (Network Weather Service) style performance estimation systems
- Supports JXTA (peer-to-peer network), JMS (Java Message Service) and more powerful native mode
- Transit time < 1 millisecond per broker
- Will have setup and broker network administration module
NaradaBrokering Naturally Supports

- Filtering of events (streams) to support different end-point requirements (e.g., PDA versus desktop, slow lines, different A/V codecs)
- Virtualization of addressing, routing, interfaces (OGSI versus pure Web services for example)
- Federation and Mediation of multiple instances of Grid services as illustrated by
  - Composition of Gridlets into full Grids (Gridlets are single computers in P2P case)
  - JXTA with peer-group forming a Gridlet
- Monitoring of messages for Service management and general autonomic functions
- Fault tolerant data transport
- Virtual Private Grid with fine-grain Security model
<table>
<thead>
<tr>
<th>Functionality</th>
<th>WebSphere MQ (formerly MQSeries)</th>
<th>Pastry</th>
<th>NaradaBrokering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of nodes hosting the messaging infrastructure</td>
<td>Medium (MQ is based on the point-to-point model. There is a limit on the effectiveness of this mode in large configurations).</td>
<td>Very large</td>
<td>Very large</td>
</tr>
<tr>
<td>JMS Compliant</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Guaranteed Messaging (Robust)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Support for routing P2P Interactions</td>
<td>No</td>
<td>Yes</td>
<td>JXTA and later Gnutella</td>
</tr>
<tr>
<td>Support for Audio/Video Conferencing &amp; raw RTP clients</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Communication through proxies and firewalls</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Support for XPath queries/ subscriptions</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>end-to-end Security</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Network Performance Monitoring</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Functionality II</td>
<td>WebSphere MQ (formerly MQSeries)</td>
<td>Pastry</td>
<td>NaradaBrokering</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------------------------------</td>
<td>-----------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Workflow Support</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Support for P2P distributed caching</td>
<td>No</td>
<td>Yes (Squirrel)</td>
<td>No</td>
</tr>
<tr>
<td>Platforms or Hosting Environments</td>
<td>35 different OS/platforms supported. Also supports the Java Platform.</td>
<td>Supported on platforms which support C# (Microsoft) or Java (Rice).</td>
<td>Platforms supporting Java 1.4 (tunneling C++)</td>
</tr>
<tr>
<td>Maturity of Software</td>
<td>Extremely mature, with very robust diagnostic information</td>
<td>Fair</td>
<td>Fair with some “production” testing</td>
</tr>
<tr>
<td>Transport Protocols Supported</td>
<td>TCP, HTTP, Multicast, SSL, SNA etc.</td>
<td>TCP, UDP</td>
<td>TCP (Blocking and non-blocking), UDP, Multicast, HTTP, SSL, RTP, (GridFTP)</td>
</tr>
<tr>
<td>Multiple transport protocols over multiple hops.</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Broker Network Design Interface</td>
<td>No</td>
<td>No</td>
<td>In Progress</td>
</tr>
</tbody>
</table>
Communication specified in terms of user goal and Quality of Service – not in choice of port number and protocol

Protocols have become overloaded e.g. MUST use UDP for A/V latency requirements but CAN’t use UDP as firewall will not support ………

A given communication can involve multiple transport protocols and multiple destinations – the latter possibly determined dynamically

![Diagram with nodes and connections representing communication protocols and filtering options]

- Communication
- Protocols
- Quality of Service
- Multiple transport protocols
- Multiple destinations
- Firewall
- UDP
- NB Brokers
- Satellite
- Fast Link
- Hand-Held Protocol
- Dial-up Filter
- Software Multicast
- Client Filtering
Grid Messaging Substrate

Standard client-server style communication.

Substrate mediated communication removes transport protocol dependence.

Any Protocols satisfying QoS

Messaging Substrate has “intelligence” “any desired fault tolerance” “logging” etc. Hope to demonstrate with multicast fault tolerant GridFTP
Applications interface to **NaradaBrokering** through **UserChannels** which NB constructs as a set of **links** between NB Brokers acting as “**waystations**” which may need to be dynamically instantiated

**UserChannels** have **publish/subscribe semantics** with **topics** labeled as in JMS, JXTA or “native XML Schema”
- Each **userchannel source** publishes to a topic
- **Destination userchannels** subscribe to topics

**Links** implement a single conventional “**data**” protocol.
- This is **store and forward** model – expensive but seems allowed as overhead small compared to network delay
- Interface to add **new transport protocols** within the Framework
- **Administrative channel** (HTTP) negotiates the best available communication protocol for each link
Different links can have different underlying transport implementations

- Implementations in the current release include support for TCP, UDP, Multicast, SSL, RTP and HTTP.

- Uses performance measurements to choose routing and protocol performance data available as a Web Service.

- Supports communication through proxies and firewalls such as iPlanet, Netscape, Apache, Microsoft ISA and Checkpoint.

- Future will include optimized PDA and FTP protocols.
Performance Monitoring

- Every broker incorporates a Monitoring service that monitors links originating from the node.
- Every link measures and exposes a set of metrics
  - Average delays, jitters, loss rates, throughput.
- Individual links can disable measurements for individual or the entire set of metrics.
- Measurement intervals can also be varied
- Monitoring Service, returns measured metrics to Performance Aggregator.
Aggregated information can be used to

- Circumvent bottlenecks
- Aid routing algorithms
- Facilitate Dynamic Load-balancing
- Diagnostics as a portlet to "administrators"

<table>
<thead>
<tr>
<th>Link</th>
<th>data rate (KB/sec)</th>
<th>avg. latency (ms)</th>
<th>min. latency (ms)</th>
<th>max. latency (ms)</th>
<th>jitter (ms)</th>
<th>message rate (msg/sec)</th>
<th>throughput (KB/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host-&gt;Sierra443 ssl/Link18213997</td>
<td>1.84</td>
<td>1.44</td>
<td>1.44</td>
<td>1.44</td>
<td>0</td>
<td>4.61</td>
<td>66.67</td>
</tr>
<tr>
<td>Host-&gt;Sierra443 ssl/Link7405914</td>
<td>1.84</td>
<td>0.93</td>
<td>0.89</td>
<td>1</td>
<td>0</td>
<td>4.61</td>
<td>66.67</td>
</tr>
<tr>
<td>Host-&gt;Sierra443 ssl/Link02603497</td>
<td>1.84</td>
<td>0.85</td>
<td>0.78</td>
<td>0.89</td>
<td>0</td>
<td>4.61</td>
<td>66.67</td>
</tr>
<tr>
<td>Host-&gt;Sierra443 ssl/Link83058044</td>
<td>1.84</td>
<td>0.78</td>
<td>0.78</td>
<td>0.78</td>
<td>0</td>
<td>4.61</td>
<td>66.67</td>
</tr>
<tr>
<td>Host-&gt;Sierra443 ssl/Link34522378</td>
<td>1.84</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4.61</td>
<td>66.67</td>
</tr>
<tr>
<td>Host-&gt;Sierra443 ssl/Link1052607</td>
<td>1.84</td>
<td>1.28</td>
<td>1.11</td>
<td>1.44</td>
<td>0</td>
<td>4.61</td>
<td>66.67</td>
</tr>
<tr>
<td>Host-&gt;Gen:443 ssl/Link7268</td>
<td>1.83</td>
<td>1.22</td>
<td>1.22</td>
<td>1.22</td>
<td>0</td>
<td>4.58</td>
<td>64.52</td>
</tr>
<tr>
<td>Host-&gt;Gen:443 ssl/Link4659</td>
<td>1.83</td>
<td>1.22</td>
<td>0.78</td>
<td>1.56</td>
<td>0</td>
<td>4.58</td>
<td>63.84</td>
</tr>
<tr>
<td>Host-&gt;Gen:443 ssl/Link8934</td>
<td>1.83</td>
<td>0.96</td>
<td>0.78</td>
<td>1.33</td>
<td>0</td>
<td>4.58</td>
<td>64.52</td>
</tr>
<tr>
<td>Host-&gt;Gen:443 ssl/Link75372</td>
<td>1.83</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4.58</td>
<td>64.52</td>
</tr>
<tr>
<td>Host-&gt;Gen:443 ssl/Link42225</td>
<td>1.83</td>
<td>1.11</td>
<td>1.11</td>
<td>1.11</td>
<td>1</td>
<td>4.58</td>
<td>64.52</td>
</tr>
<tr>
<td>Host-&gt;Gen:443 ssl/Link03112</td>
<td>1.83</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4.58</td>
<td>64.52</td>
</tr>
</tbody>
</table>
Mean transit delay for message samples in NaradaBrokering: Different communication hops

Pentium-3, 1GHz,
256 MB RAM
100 Mbps LAN
JRE 1.3 Linux
Standard Deviation for message samples in NaradaBrokering
Different communication hops - Internal Machines

- hop-2
- hop-3
- hop-5
- hop-7

Message Payload Size (Bytes)

Standard Deviation (Milliseconds)
Architecture of Message Layer

- Need to optimize not only routing of particular messages but classic publish/subscribe problem of integrating different requests with related topics (subscribe to sports/basketball/lakers and sports)

- Related to Akamai, AOL … caching and Server optimization problem

Hypercube of NB Brokers (logical not physical)

N≈100 clients per leaf node for A/V conferencing

Scale using NB broker network
Organization of Profiles and Routing

- **End-point subscriptions** are stored hierarchically within the system as part of their profile.
  - A broker maintains end-point subscriptions, cluster-controller maintains broker subscriptions and so on.
  - Profile is set of constraints such as `Session=PVMMPI-1`

- When an event is received, the event is **matched** against stored profiles and destinations are computed.
  - A **cluster-controller computes broker destinations**.
  - A **broker computes end-point destinations**.

- Every broker node, when supplied with a set of destinations, computes the **minimum broker-hops** to take to reach these destinations.
  - Takes accounts of failed brokers and bad links.
NaradaBrokering: Matching Engines

Matching engines are responsible for matching events to stored profiles of subscribing end-points.

NaradaBrokering supports a variety of topic-event matching engines supporting

- Classic “/” separated String based topics (//Sports/baseball/cubs etc.)
- Equality based <property=value> topics
- Integer labeled topics (for highest performance in A/V conferencing)
- SQL (for Java Message Service emulation)
- XPath based queries (Most powerful “Native” NB mode)
- Stand alone process
- Pentium-3 1 GHZ 256MB RAM
- JRE 1.4
- Only done once per topic
Current best test of NB Emulate JMS in multi-topic Anabas collaboration system Includes large shared display messages Up to 30 clients run regularly
Transit Delays for Message Samples in Narada and SonicMQ

**Low Rate; Small Messages**

Narada

SonicMQ (commercial JMS)
Native NaradaBrokering

- **SC03 release of NaradaBrokering** will have a high functionality native mode that provides for several features pertaining to the delivery of events between communicating end points
  - **Reliable Delivery**
  - **Ordered Delivery** (Publisher, Causal or Time)
  - **Secure delivery**
  - **Compression, Fragmentation** and partial (Sliced) delivery of content payloads.

- **End-points** can be services or user interface clients

- Availability of new **full function end-point interfaces** to communicate and interact with the broker network and other registered clients
  - **Current** release only has full function broker and simple end-points supporting **JMS and JXTA**
**End-point Services**

- Allows you to create **Consumers** (subscribers) of events (an event is a time stamped message where time stamp can be empty!)
- Allows you to create **Producers** of events (publishers)
- Allows you to **discover brokers and initialize communications with the broker**.

**Services available at the client side will perform**

- **Compression** of payloads
- **Computation of Message digests** for Integrity
- **Secure encryption** of payload based on the specified keys
- **Fragmentation** of large payloads into smaller packets
- **Redundancy** service which maintains active (alternate) connections to multiple brokers.
Event Consumer Capabilities

- Allow you to subscribe to events that conform to a certain template.
  - The specified subscription profile could topic-based strings, XPath queries, <tag=value> pairs or integer topics.

- Event Consumers can also create **Consumer constraints** to specify various properties regarding the delivery of events.

- Consumer constraints are different from subscriptions.
  - Subscriptions (or Profiles) are evaluated in a distributed fashion by the broker network,
  - Consumer constraints are QoS related and are managed by the QoS services running on the end-point.

- Consumer constraints can specify
  - Reliable Delivery of events
  - Ordered (Publisher, causal and time ordered) delivery of events
  - Exactly once delivery of events
  - Delivery after un-compression of compressed payload
  - Delivery after decrypting encrypted payload
Event Producer Capabilities

- Facilitate the generation of events in correct format (next slide)
- Facilitate the publishing of events to brokers
- Allow the creation of Publisher constraints which facilitate specification of properties that need to be satisfied by published events
- Among the constraints that can be specified include
  - Method of Securing message payloads
  - Computing message digests
  - Compressing message payloads
  - Fragmenting large payloads
Native NaradaBrokering Event

The event comprises of:

- Event headers
- **Content Synopsis** (for selection as in JMS properties WITHOUT reading body)
- Content Payload
- **Dissemination Traces** (generated on the fly as event traverses broker network)

This is different from structure of JMS or JXTA events.

This NBEvent structure supports the extra capabilities discussed earlier.

The event headers specify information regarding:

- **Security and Integrity** of encapsulated payload
- Fragmentation of events
- Compression of payloads
- **Correlation identifiers** (to define ordering between different streams as is needed in some collaboration applications)
- Priority
- Application Type
- Event Identifiers
Based on **hybrid proxy** that acts as both **Rendezvous peer** (JXTA routers) and NaradaBrokering end-point.

- **No changes to JXTA core** or constraints on interactions
  - Change made to Rendezvous layer

- **Peers are not aware** that they interact with a Narada-JXTA proxy or Rendezvous peer.

- NB provides JXTA guaranteed **long distance delivery**
- NB **federates** multiple JXTA Peer Groups
NaradaBrokering-JXTA Proxy

- **Glean** relevant information from JXTA interactions.
  - Peer group *advertisements* (XML Doc describing resource)
  - Requests/Responses to be part of peer group.
  - Messages sent to a peer group.
  - Queries and responses to these queries.

- **Subscribe** to relevant topics to ensure delivery

- **Construct** corresponding Narada-JXTA event from interactions.
  - These events lend themselves to efficient routing.
  - Duplicate events found in P2P are removed
NaradaBrokering P2P Federation

- **Applications**
  - Integrated NaradaBrokering-JXTA environment tested under *JXTA shell* and *myJxta* (InstantP2P)
  - Allow JXTA access to *JMS* Applications such as Anabas Web collaboration and distance education system
  - Can similarly federate Limewire (*Gnutella*) but postponed due to resource limitations

- **Experimental Setup**
  - Sender/receiver - (Pentium-3, 1 GHz, 256 MB RAM).
  - Every node (broker/router) hosted on a different machine (Pentium-3, 1 GHz, 256 MB RAM).
  - Machines reside on a 100 Mbps LAN
  - Run-time environment for all the processes is JDK-1.3 build Blackdown-1.3.1, Red Hat Linux 7.3
Transit delay for message samples in Narada, JXTA and Narada-JXTA
Topology - III (8 routers) Internal Machines

NaradaBr
Pure JXTA
Narada-JXTA

Transit delay for message samples in Narada, JXTA and Narada-JXTA
Topology - III (8 routers) Internal Machines

NaradaBr
Pure JXTA
Narada-JXTA

Transit Delay (Milliseconds)

Message Payload Size (Bytes)
NaradaBrokering: Security Framework

• Based on Message Level Security
  – Messages organized into topics
  – Each topic has a separate key
  – Topics can be organized into sessions

• Authentication – Confirm whether a user is really who he says he is.

• Authorization – Determine if the user is authorized to send or receive certain events

• For collaboration applications Global-MMCS (discussed later) is responsible for setting up sessions and topics with security characteristics defined in XGSP

• Digital Signing – Have the ability to verify the source of the event and whether the source is authorized to publish events conforming to the specified template.
Key Management Center (KMC) | Broker Node | Entity (Publisher or Subscriber) | SSL encrypted communications

1. Request permission to publish
2. Respond back with topic key if authorized to publish
3. Encrypt message with topic key
4. Compute Message Digest (MD)
5. Sign MD and message ID
6. Publish Message

Verify Signature & Permissions
Check integrity by verifying MD
Check ID for replay attacks

7. Request permission to subscribe
8. Respond back with topic key if authorized to subscribe
9. Create subscription request
10. Compute Message Digest
11. Sign MD and message ID
12. Issue Subscription request Message

Verify Signature
Verify Permissions for Subscribing
Check integrity by verifying MD
Check ID for replay attacks
NaradaBrokering and Collaboration

- In a Web Service view of application (as opposed to service), replication of messages into or out of a service is all you need for rich collaboration model
  - Motivates building (all) applications as Web Services
  - SVG and less completely PowerPoint as examples

- Media Servers use NB as a stream (pipe) handler
  - 100 video streams per Linux server
  - Convert codecs as a filter service

- Network QoS is dominant problem in widespread use of internet collaboration
  - NB’s communication virtualization and performance-based autonomic delivery tries to address
Collaboration and Web Services

- **Collaboration** has
  - a) Mechanism to set up members (people, devices) of a “collaborative sessions”
  - b) Shared generic tools such as text chat, white boards, audio-video conferencing
  - c) Shared applications such as Web Pages, PowerPoint, Visualization, maps, (medical) instruments ….

- b) and c) are “just shared objects” where objects could be Web Services but rarely are at moment
  - We can port objects to Web Services and build a general approach for making Web services collaborative

- a) is a “Service” which is set up in many different ways (H323 SIP JXTA are standards supported by multiple implementations) – we should make it a WS
Shared Event Collaboration

- All collaboration is about **sharing events defining state changes**
  - **Audio/Video conferencing** shares events specifying in compressed form audio or video
  - **Shared display** shares events corresponding to change in pixels of a frame buffer
  - **Instant Messengers** share updates to text message streams
  - **Microsoft events** for shared PowerPoint (file replicated between clients) as in Access Grid

- **Finite State Change NOT Finite State Machine architecture**

- **Using Web services** allows one to expose updates of all kinds as messages
  - “Event service” for collaboration is similar to Grid notification service and we effectively define SDE’s (service data elements) in OGSI

- **Group (Session) communication service** is needed for the delivery of the update events
  - Using Event Messaging middleware makes messaging universal
Shared Output Port Collaboration

Web Service Message Interceptor

Collaboration as a WS
Set up Session with XGSP

Master

WS Viewer
WS Display

Other Participants

WS Viewer
WS Display

Event (Message) Service

Text Chat
Whiteboard
Multiple masters

Web Service
Application or Content source

WSDL

Shared Output Port Collaboration
Shared Input Port (Replicated WS) Collaboration

Collaboration as a WS
Set up Session with XGSP

Master

Event (Message) Service

Web Service

Other Participants

Web Service

Web Service

WS Viewer

WS Display

WS Viewer

WS Display
Global-MMCS 2.0 XGSP based MCU

- We are building an open source protocol independent Web Service “MCU” which will scale to an arbitrary number of users and provide integrated thousands of simultaneous users collaboration services.
- We will deploy it globally and hope to test with later this year.
- The function of A/V media server will be distributed using NaradaBrokering architecture.
  - Media Servers mix and convert A/V streams
- Open Global-MMCS MCU based on the following open source projects
  - openh323 is basis of H323 Gateway
  - NIST SIP stack is basis of SIP Gateway
  - NaradaBrokering is open source messaging from Indiana
  - Java Media Framework basis of Media Servers
XGSP Web Service MCU Architecture

Use Multiple Media servers to scale to many codecs and many versions of audio/video mixing

- Session Server
- XGSP-based Control
- Media Servers
- Filters
- Web Services
- NaradaBrokering
- All Messaging
- NB Scales as distributed
- High Performance (RTP)
- and XML/SOAP and ..

Gateways convert to uniform XGSP Messaging

NaradaBrokering

Admire
SIP
H323
Access Grid
Native XGSP
Polycom, Access Grid and RealVideo views of multiple streams using A/V Web Service integrating SIP and H323
We conducted extensive performance tests on audio and video servers.

**Video:**
- The test shows that our video server is capable of supporting 100-300 clients if there is only one video sender.
- Video Server Machine: 1.2GHz Intel Pentium III dual CPU, 1GB MEM, RedHat Linux 7.3

**Audio:**
- Our tests show that audio server can support 5 concurrent sessions (250 participants in total) without any packet droppings.
- Audio Server Machine: 2.5GHz Pentium 4 CPU, 512MB memory, Windows XP machine

**Scale** with logarithmic Broker network
Average delays per packet for 50 video-clients
NaradaBrokering Avg=2.23 ms, JMF Avg=3.08 ms

NaradaBrokering-RTP
JMF-RTP
Average jitter (std. dev) for 50 video clients.

NaradaBrokering Avg=0.95 ms, JMF Avg=1.10 ms
Comparison between the performance of NaradaBrokering and JMF

Average delays/packet for 12 (of the 400 total) video-clients. NaradaBrokering Avg=80.76 ms, JMF Avg=229.23 ms
Comparison between the performance of NaradaBrokering and JMF

Average jitter/packet for 12 (of the 400 total) video clients.
NaradaBrokering Avg=13.38 ms, JMF Avg=15.55 ms
Application Web Services and Universal Access

- NaradaBrokering can link lightweight clients (V in MVC) to Web Services holding as a Web service the “guts of an application” (M in MVC)
  - This allows customizable user interfaces gotten by mapping between client profile protocols at NB
  - Supports collaboration between diverse clients
Collaborative SVG Web Service

- **SVG** is W3C 2D Vector Graphics standard and is interesting for visualization and as a simple PowerPoint like application
  - Further SVG is built on **W3C DOM** and one can generalize results to all W3C DOM-based applications (“all” in future?)
- **Apache Batik** SVG is **Java** and **open source** and so it is practical to modify it to explore
  - **Real Applications as a Web Service**
  - **Collaboration as a Web Service**
  - **MVC model** and web services with implications for **portlets**
- We use **NaradaBrokering** and **Global-MMCS** to control collaboration; support PDA Cell-phone and desktop clients; are restructuring Batik as **MVC Web Service**
  - Good progress in all areas see
  - [http://www.svgarena.org](http://www.svgarena.org) for SVG Games
Interrupts in traditional monolithic applications become “real messages” not directly method calls
Natural for collaboration and universal access
Collaborative SVG As A Web Service

Control flow for collaboration

NaradaBrokering

From Collaboration

From Master

Participating Client
Collaborative SVG Chess Game in Batik Browser
Integration of PDA, Cell phone and Desktop Grid Access
InterGrids Federated Grid using NB

- Build a P2P Network where each component (cell or Gridlet) is itself a Grid
- If cell is a single computer, reduces to using NB to build communication infrastructure between nodes of P2P network
- If cell is a JXTA peer group, then InterGrids includes previous federation of JXTA Peer Groups
InterGrids Mediation Architecture

- **NB** acts as a *Mediation agent* in such a *Cellular Grid*
- Using federated security model constructs a VPN like *Virtual Private Grid*
- *Mediation* includes more than routing (as in current JXTA) as can map between Interface standards
- Each *Gridlet* can use different *Service* standards
- *Services register* interfaces with mediator giving ways to map using perhaps *OGSA* as a common intermediate form
- Allows integration of *OGSI* and “pure web service” or *Jini* or *JXTA* based Grids where each Grid uses its natural service architecture
- Support *interoperable* (like Job Submission) and *federated* (like registry or metadata catalog) services
- Exploits *stream filtering* capability of *NB*
Autonomic Services in InterGrids

- In a Web (Grid) Service architecture, the state of any service is defined by its initial condition and all the messages (including ordering) that it receives
  - This how shared event model of collaboration works
- This is a “Finite State Change” model analogous to saving file and “undo” command in many editors
- NB plus a robust store can “guarantee” to save all these messages for (all) services
- This allows one to build both "autonomic data transport" and "autonomic services" since these services can sustain packet losses in transport and can also sustain failures of apps/brokers
  - archived messages (previous invocations, published events etc) can be retransmitted to reconstruct state at the service or to correct a transport error.
- Anomalies in message traffic (such as a publisher or subscriber are silent) can be detected by NB and signal problems
- We are building examples of both scenarios using GridFTP as our data transport example
- We will build a sample autonomic visualization service with detection of failed servers and brokers
Conclusions

The factor of 105 in communication latency has allowed a very sophisticated Grid Messaging system. It took several years to understand how to design and build PVM and MPI; it will take longer for the Grid?

We can virtualize interfaces and communication and so allow easier system integration

**NB** is research project and needs a lot of testing

**InterGrids** very primitive

**NB** needs a model for “stable storage” to interface with its reliable message delivery
  - Support both intrinsically fault tolerant stores and replicated stores linked to replicated brokers

What stream handling in **NB** could go in either network routers or parallel computing messaging and I/O?

Download it from [http://www.naradabrokering.org](http://www.naradabrokering.org)

Many new nifty capabilities for SC03 release