Managing Grid and Web Services and their exchanged messages

Authors

Harshawardhan Gadgil (his PhD topic), Geoffrey Fox, Shrideep Pallickara, Marlon Pierce

Community Grids Lab, Indiana University

Presented by

Geoffrey Fox

`gcf@indiana.edu`

Talk Outline

- Management (Configuration/QoS Implementation)
- Existing Management Approaches
- Overview of NaradaBrokering Messaging infrastructure
- NaradaBrokering Use Cases
- Our Management Architecture
- Management Prototype and some results from it
- Future work and conclusion

Management I

- Characteristics of today’s (Grid) applications
  - Increasing complexity
  - Components widely dispersed and disparate in nature and access
    - Span different administrative domains
    - Under differing network / security policies
    - Limited access to resources due to presence of firewalls, NATs etc… (major focus in prototype)
  - Dynamic
    - Components (Nodes, network, processes) may fail
- Services must meet
  - General QoS and Life-cycle features
  - (User defined) Application specific criteria
- Need to “manage” services to provide these capabilities

Management II

- Management Operations* include
  - Configuration and Lifecycle operations (CREATE, DELETE)
  - Handle RUNTIME events
  - Monitor status and performance
  - Maintain system state (according to user defined criteria)
- Protocols like WS-Management/WS-DM define inter-service negotiation and how to transfer metadata
- We are designing/prototyping a system that will manage a general world wide collection of services and their network links
- We are starting with our messaging infrastructure as
  - we need this to be robust in Grids we are using it in (Sensor and amterial science)
  - we are using it in management system
  - and it has critical network requirements

* From WS – Distributed Management

http://devresource.hp.com/drc/slide_presentations/wsdm/index.jsp

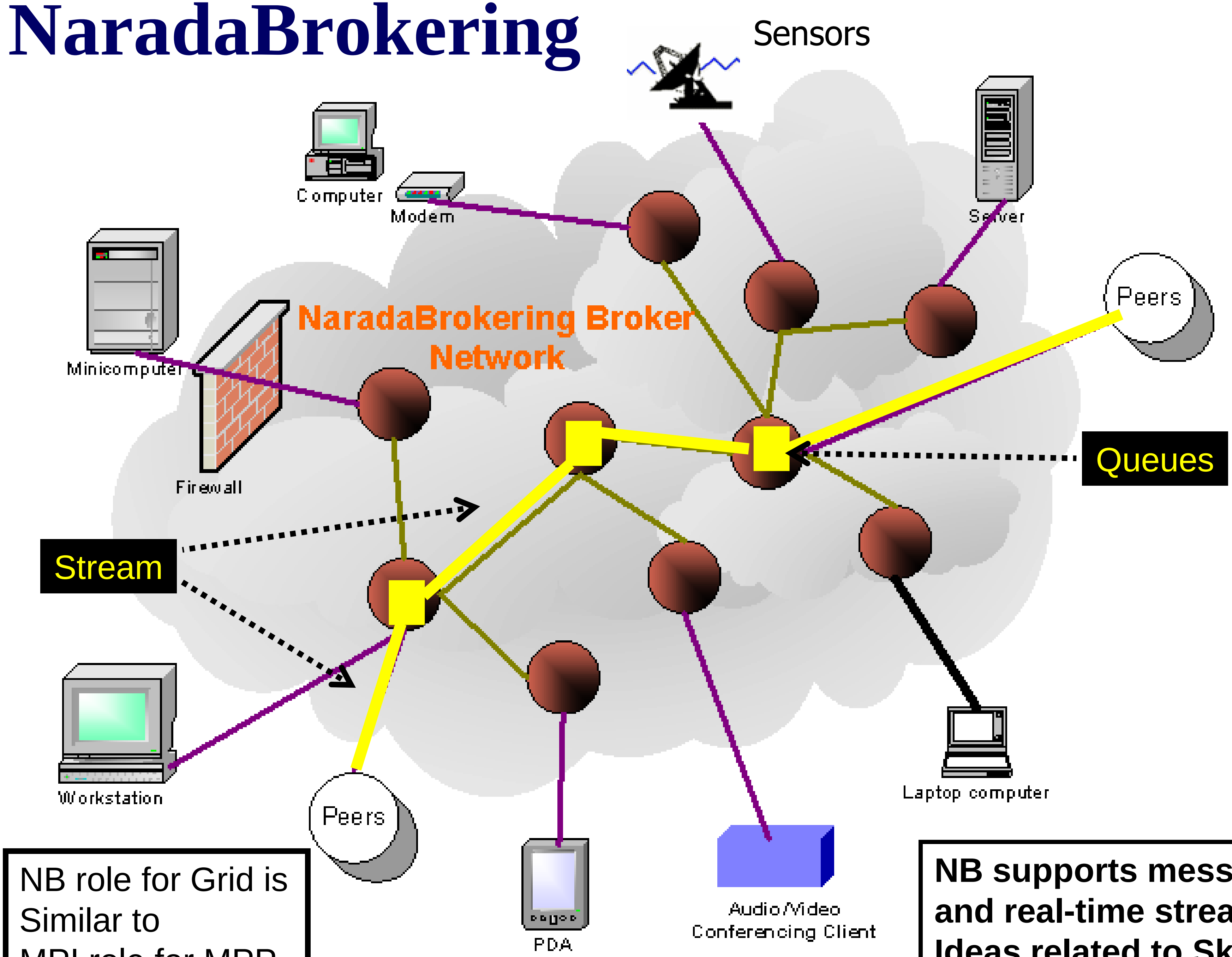
Some Existing Management Protocols I

- SNMP – Simple Network Management Protocol
  - Application layer protocol, based on reliable connection-oriented protocol.
  - Enables network admins to manage network performance and find and solve problems
  - Lacks security (Authentication), hence vulnerable to masquerading occurences, information modification etc…
  - In most cases "SET" operation not implemented and hence degenerates to monitoring facility only.
  - Mainly deals with hardware resources.
- CMIP – Common Management Information services and Protocols
  - Provides improved security (access control, authorization and security logs) over SNMP

Some Existing Management Protocols II

- CIM – Common Information Management
  - Object oriented model that represents and organizes information
  - Allows extending existing management standards
- Web Service based (WS-Management, WS-Distributed Management), Upcoming merger to a common Web Service based management architecture
  - Helps making management interoperable
  - Work underway to map CIM constructs to WSDM
  - Provides a SOAP binding for various verbs (CREATE / DELETE / SET / GET)
  - Defines negotiation between services and some metadata
  - Manage services or non-service entities wrapped as services
    - HARDWARE – Processors, printers etc…
    - SOFTWARE – Processes (E.g. Brokers in our case)
    - Managers are ALWAYS Services

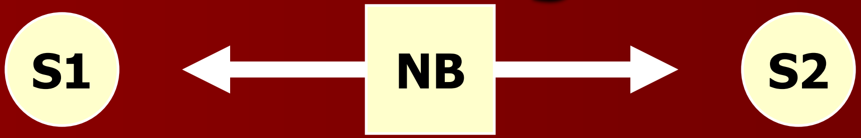
NaradaBrokering



NB role for Grid is
Similar to
MPI role for MPP

NB supports messages and real-time streams
Ideas related to Skype

NaradaBrokering Core Features I



- Supports general linkage of threads, processes and services
- Message-level security (See Grid06 paper http://www.naradabrokering.org/papers/NB-Security.pdf)
- Message Payload Options
  - Compression and Decompression of payloads
  - Fragmentation and Coalescing of payloads
- Message Compliance
  - Java Message Service (JMS) 1.0.2b compliant
  - (Obsolete) Support for routing P2P JXTA interactions.
- Grid Feature Support
  - NaradaBrokering enhanced Grid-FTP. (Old) bridge to Globus.
- Web Service Support
  - Implementations of WS-ReliableMessaging, WS-Reliability and WS-Eventing.

NaradaBrokering Core Features II

- Multiple protocol transport support
  - Transport protocols supported include TCP, Parallel TCP streams, UDP, Multicast, SSL, HTTP and HTTPS.
  - Communications through authenticating proxies/firewalls & NATs. Network QoS based Routing
  - Allows Highest performance transport with 1-2 ms overhead and <1ms timing guarantees
- Subscription Formats
  - Subscription can be Strings, Integers, XPath queries, Regular Expressions, SQL and tag=value pairs.
- Reliable Delivery
  - Robust and exactly-once delivery in presence of failures
- Ordered Delivery
  - Producer Order and Total Order over a message type. Time Ordered delivery using Grid-wide NTP based absolute time
- Recovery & Replay
  - Recovery from failures and disconnects. Replay of events/messages at any time. Buffering services.
- Open Source http://www.naradabrokering.org

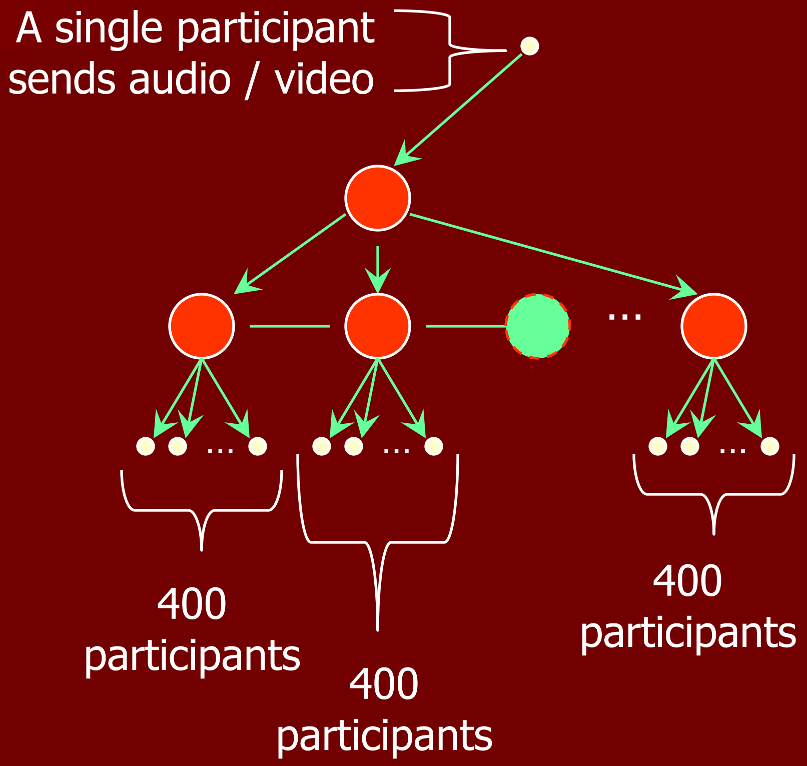
NaradaBrokering Management Needs

- NaradaBrokering Distributed Messaging System consists of peers (brokers) that collectively form a scalable messaging substrate. Optimizations and configurations include:
  - Where should brokers be placed and how should they be connected, E.g. RING, BUS, TREE, HYPERCUBE etc…, each TOPOLOGY has varying degree of resource utilization, routing, cost and fault-tolerance characteristics.
- Static topologies or topologies created using static rules may be inefficient in some cases
  - E.g., In CAN, Chord a new incoming peer randomly joins nodes in the network. Network distances are not taken into account and hence some lookup queries may span entire diameter of network
  - Runtime metrics provide dynamic hints on improving routing which leads to redeployment of messaging system (possibly) using a different configuration and topology
  - Can use (dynamically) optimized protocols (UDP v TCP v Parallel TCP) and go through firewalls but no good way to make choices dynamically
- These actions collectively termed as Managing the Messaging Middleware

NaradaBrokering Use Cases

- Use case I: Audio – Video Conferencing (GlobalMMCS project, http://www.globalmmcs.org) which uses NaradaBrokering as a event delivery substrate
- Consider a scenario where there is a teacher and 10,000 students. One would typically form a TREE shaped hierarchy of brokers
- One broker can support up to 400 simultaneous video clients and 1500 simultaneous audio clients with acceptable quality*. So one would need (10000 / 400 ≈ 25 broker nodes).
- May also require additional links between brokers for fault-tolerance purposes
- Use Case II: Sensor Network
- Both use cases need high QoS streams of messages
- Use Case III: Management System itself

Use Case I



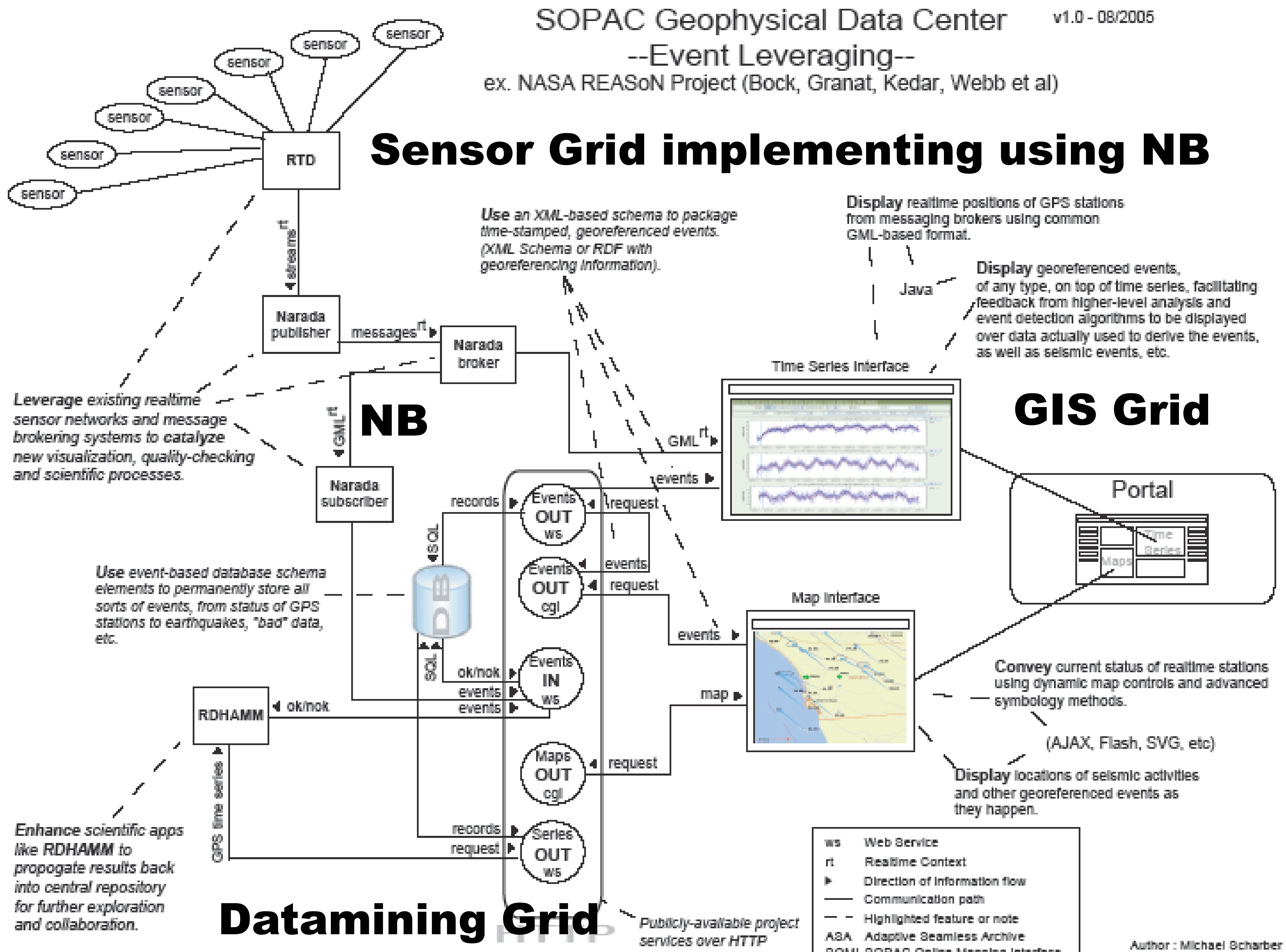
* *"Scalable Service Oriented Architecture for Audio/Video Conferencing"*, Ahmet Uyar, Ph.D. Thesis, May 2005

Sensor Grid implementing using NB

NB

GIS Grid

Datamining Grid



SOPAC Geophysical Data Center

--Event Leveraging--

ex. NASA REASoN Project (Bock, Granat, Kedar, Webb et al)

v1.0 - 08/2005

Use an XML-based schema to package time-stamped, georeferenced events. (XML Schema or RDF with georeferencing information).

Display realtime positions of GPS stations from messaging brokers using common GML-based format.

Display georeferenced events, of any type, on top of time series, facilitating feedback from higher-level analysis and event detection algorithms to be displayed over data actually used to derive the events, as well as seismic events, etc.

Leverage existing realtime sensor networks and message brokering systems to catalyze new visualization, quality-checking and scientific processes.

Use event-based database schema elements to permanently store all sorts of events, from status of GPS stations to earthquakes, "bad" data, etc.

Convey current status of realtime stations using dynamic map controls and advanced symbology methods.

(AJAX, Flash, SVG, etc)

Display locations of seismic activities and other georeferenced events as they happen.

Enhance scientific apps like RDHAMM to propogate results back into central repository for further exploration and collaboration.

Publicly-available project services over HTTP

ws Web Service
rt Realtime Context
▶ Direction of information flow
—— Communication path
— — Highlighted feature or note
ASA Adaptive Seamless Archive
SOMI SOPAC Online Mapping Interface

Author : Michael Scharber

Core Features of Management Architecture

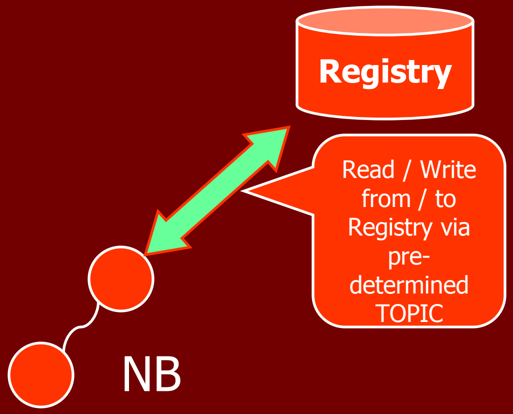
- Remote Management
  - Allow management irrespective of the location of the resource (as long as that resource is reachable via some means)
- Traverse firewalls and NATs
  - Firewalls complicate management by disabling access to some transports and access to internal resources
  - Utilize tunneling capabilities and multi-protocol support of messaging infrastructure
- Extensible
  - Management capabilities evolve with time. We use a service oriented architecture to provide extensibility and interoperability
- Scalable
  - Management architecture should be scale as number of managees increases
- Fault-tolerant
  - Management itself must be fault-tolerant. Failure of transports OR management components should not cause management architecture to fail.

Management System built in terms of

- Bootstrap System – Robust itself by Replication
- Registry for metadata (distributed database) – Robust by standard database techniques and our system itself for Service Interfaces
- NaradaBrokering for robust tunneled messages – NB itself robust using our system
- Managers – Easy to make robust using our system
- Managees – what you are managing – Our system makes robust – There is NO assumption that Managed system uses NB

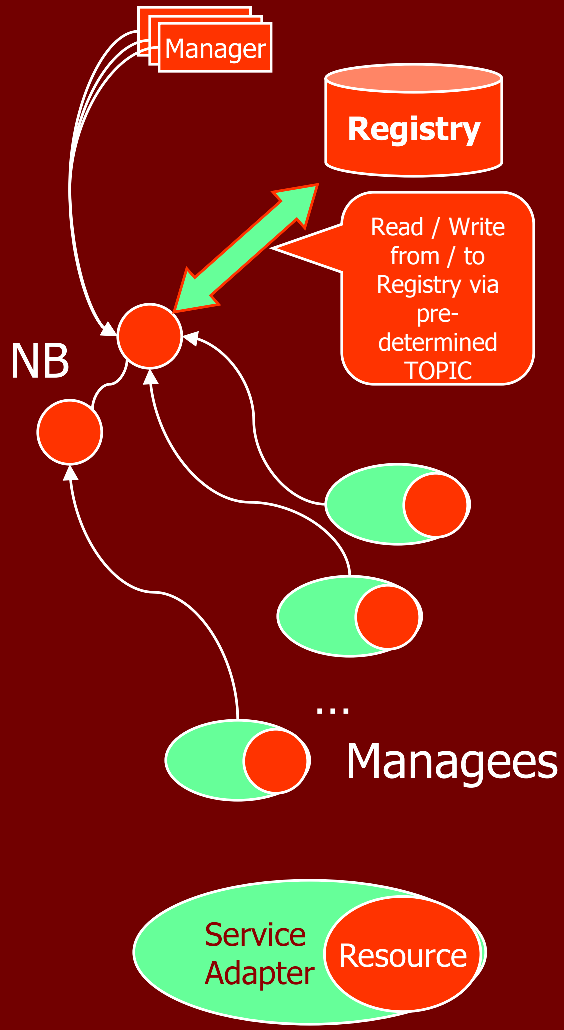
Basic Management Architecture I

- Registry
  - Stores system state.
  - Fault-tolerant through replication
  - Could be a global registry OR separate registries for each domain (later slide)
  - Current implementation uses a simple in-memory system
  - Will use our WS - Context service as our registry (Service/Message Interface to in-memory JavaSpaces cache and MySQL)
  - Note metadata transported by messages but we use distributed database to implement
- Messaging Nodes
  - NaradaBrokering nodes that form a scalable messaging substrate
  - Main purpose is to serve as a message delivery mechanism between Managers and Service Adapters (Managees) in presence of varying network conditions



Basic Management Architecture II

- Resources to Manage (Managee)
  - If the resources DO NOT have a Web Service interface, we create a Service Adapter (a proxy that provides the Web Service interface as a wrapper over the basic management functionality of the resource).
  - The Service Adapters connect to existing messaging nodes. This mainly leverages multi-protocol transport support in the messaging substrate. Thus, alternate protocols may be used when network policies cause connection failures
- Managers
  - Active entities that manage the resources.
  - May be multi-threaded to improve scalability (currently under further investigation)



Architecture

Use of Messaging Nodes

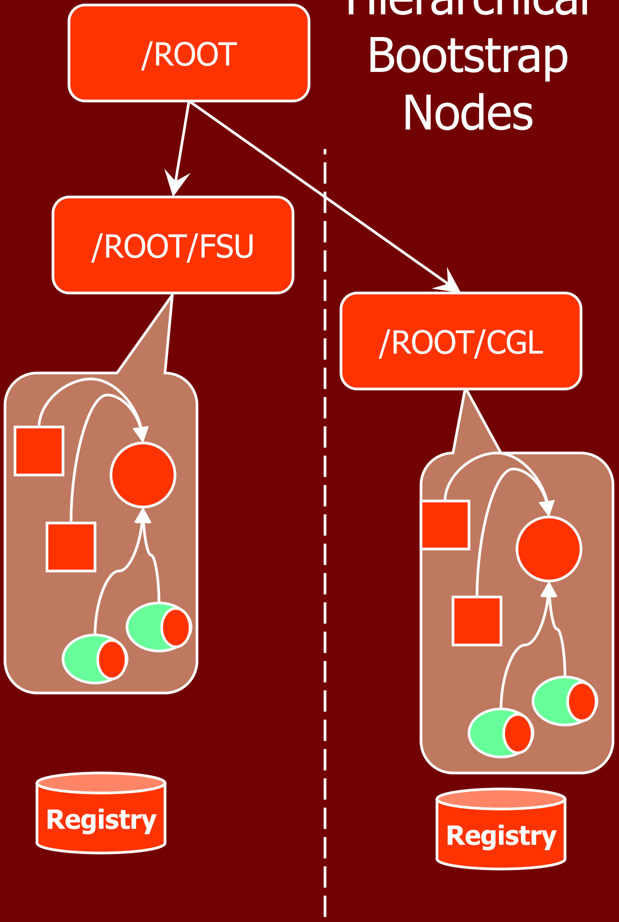
- Service adapters and Managers communicate through messaging nodes
- Direct connection possible, however
  - This assumes that the service adapters are appropriately accessible from the machines where managers would run
    - May require special configuration in routers / firewalls
  - Typically managers and messaging nodes and registries are always in the same domain OR a higher level network domain with respect to service adapters
- Messaging Nodes (NaradaBrokering Brokers) provides
  - A scalable messaging substrate
  - Robust delivery of messages
  - Secure end-to-end delivery

Architecture

Bootstrapping Process

- The architecture is arranged hierarchically.
  - Resources in different domains can be managed with separate policies for each domain
- A Bootstrapping service is run in every domain where the management architecture exists.
  - Serves to ensure that the child domain bootstrap process are always up and running.
    - Periodic heartbeats convey status of bootstrap service
  - Bootstrap service periodically spawns a health-check manager that checks health of the system (ensures that the registry and messaging nodes are up and running and that there are enough managers for managees)
    - Currently 1 manager per managee

Hierarchical Bootstrap Nodes



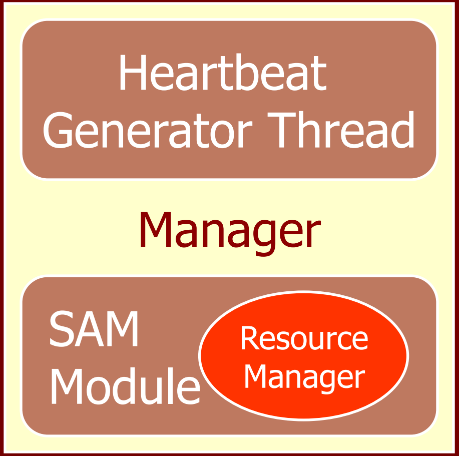
Architecture: User Component

- Application-specific specification of the characteristics that the resources/services being managed, should maintain.
  - Impacts Managee interface, registry and Manager
- Generic and Application specific policies are written to the registry where it will be picked up by a manager process.
- Updates to the characteristics (WS-Policy in future) are determined by the user.
- Events generated by the Managees are handled by the manager.
  - Event processing is determined by policy (future work),
    - E.g. Wait for user's decision on handling specific conditions
    - The event can be processed locally, so execute default policy, etc…
- Note Managers will set up services if registry indicates that is appropriate; so writing information to registry can be used to start up a set of services

Architecture

Structure of Managers

- Manager process starts appropriate manager thread for the manageable resource in question
  - Heartbeat thread periodically registers the Manager in registry
  - SAM (Service Adapter Manager) Module Thread starts a Service/Resource Specific “Resource Manager” that handles the actual management task
  - Management system can be extended by writing ResourceManagers for each type of Managee



Prototype

- We illustrate the architecture by managing the distributed messaging middleware, NaradaBrokering as illustrated by 3 use cases
  - This example motivated by the presence of large number of dynamic peers (brokers) that need configuration and deployment in specific topologies
- Use WS – Management (June 2005) parts (WS – Transfer [Sep 2004], WS – Enumeration [Sep 2004] and WS – Eventing) (could use WS-DM)
  - WS – Enumeration implemented but we do not foresee any immediate use in managing the brokering system
  - WS – Transfer provides verbs (GET / PUT / CREATE / DELETE) which allow us to model setting and querying broker configuration, instantiating brokers and creating links between them and finally deleting brokers (tear down broker network) and re-deploy with possibly a different configuration and topology
  - WS – Eventing (will be leveraged from the WS – Eventing capability implemented in OMII)
- WS – Addressing [Aug 2004] and SOAP v 1.2 used (needed for WS-Management)
  - Used XmlBeans 2.0.0 for manipulating XML in custom container.
- WS-Context will replace current registry

Prototype Components

- Broker Service Adapter
  - Note NB illustrates an electronic entity that didn’t start off with an administrative Service interface
  - So add wrapper over the basic NB BrokerNode object that provides WS – Management front-end
  - Also provides a buffering service to buffer undeliverable responses
    - These will be retrieved later by a separate Request – Response message exchange
- Broker Network Manager
  - WS – Management client component that is used to configure a broker object through the Broker Service Adapter
  - Contains a Request-Response as well as Asynchronous messaging style capabilities
  - Contains a topology generator component that determines the wiring between brokers (links that form a specific topology)
    - For the purpose of prototype we simply create a CHAIN topology where each i^{th} broker is connected to $(i-1)^{st}$ broker

Prototype

Resources/Properties Modeled (very specific to NaradaBrokering)

Resource URI	Operations	Description
BROKER	Create Delete	Instantiates the broker with current configuration Deletes the broker node
LINK (Note we manage brokers and streams)	Create Delete	Creates a link between two brokers Deletes the link between two brokers
CONFIGURATION, CONFIGURATION PROPERTY	Get Put	Retrieves the current configuration / a single property Saves the specified configuration / single property
NODE ADDRESS, GATEWAY ADDRESS	Create	Assigns a NODE / GATEWAY address to the current node if one is not already assigned

Benchmarks - I

- Test -I: Deployed a network of 8 brokers on 8 different machines.
  - Noted the overhead (Create Message + Marshall SOAP + Network Latency + Unmarshall SOAP) introduced by the system
    - Set Configuration: 73.56 mSec
    - Get Configuration: 61.11 mSec
    - Create Broker: 61.4 mSec
    - Create Link: 88.35 mSec
    - Get Node Address: 68.94 mSec
    - Delete Broker: 75.02 mSec
  - Used direct HTTP connection (custom written SOAP client / server to allow for use of SOAP 1.2 based messages and provide compatibility with other software)
  - Currently working on detailed analysis of benchmarks with time probably largely determined by marshalling and un-marshalling messages which are one-way Web Service invocations over TCP

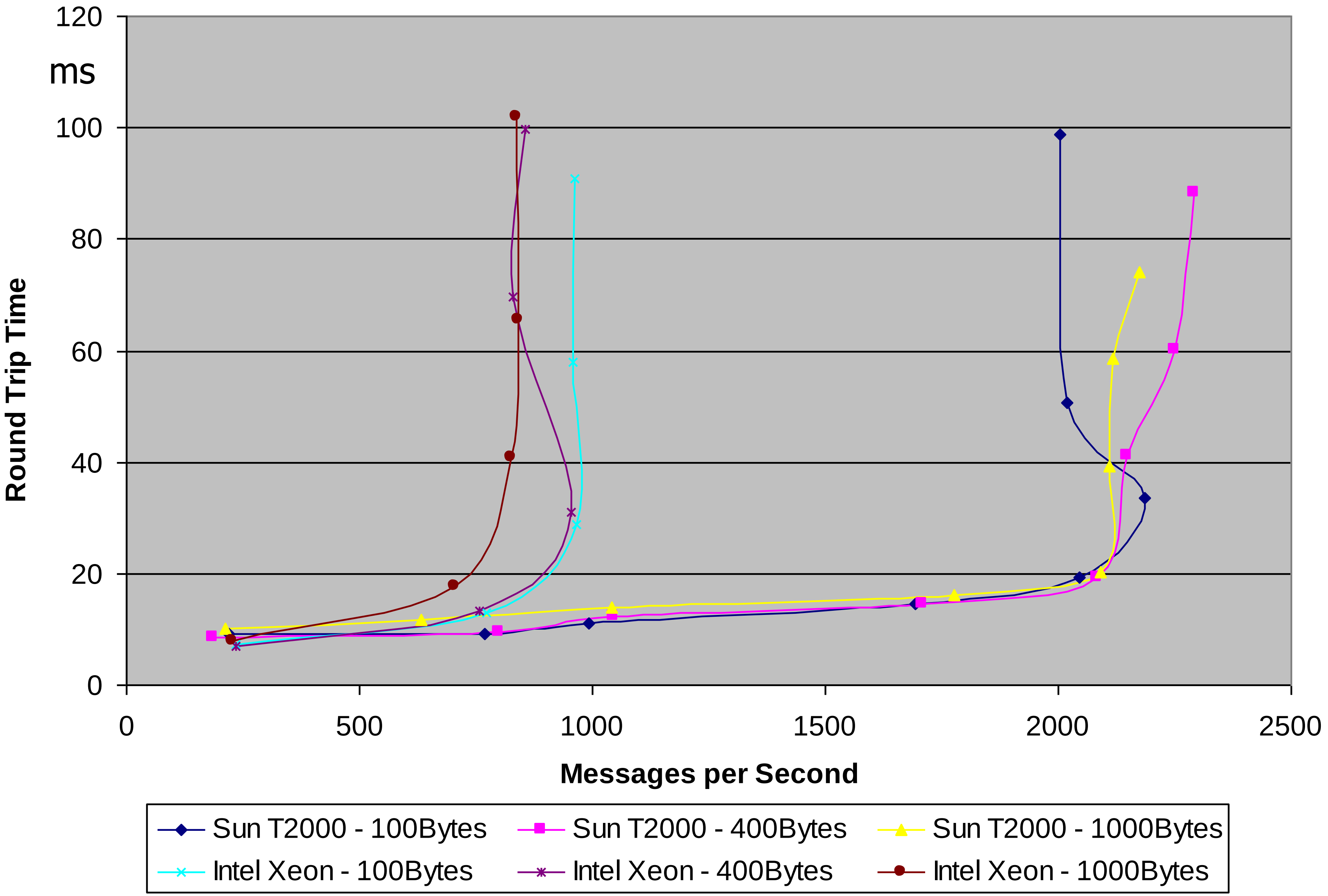
Benchmarks - II

- Test -II: Managed brokers were present on remote machines
  - 3 brokers behind a Home DSL ROUTER
  - Used a distributed messaging substrate to route messages to appropriate recipients.
    - Distributed messaging substrate provides multiple transport support, tunneling through firewalls (to enable remote management)
  - Noted the overhead introduced by the system
    - Set Configuration: 172.01 mSec
    - Get Configuration: 178.69 mSec
    - Create Broker: 149.52 mSec
    - Create Link: 143 mSec
    - Get Node Address: 144.34 mSec
    - Delete Broker: 131.41 mSec
  - SOAP message is received by a HTTP mapper service and is relayed to the service adapter by publishing a message over a pre-determined topic (with the soap message as event payload). Response is relayed back in a similar fashion.

Benchmarks - III

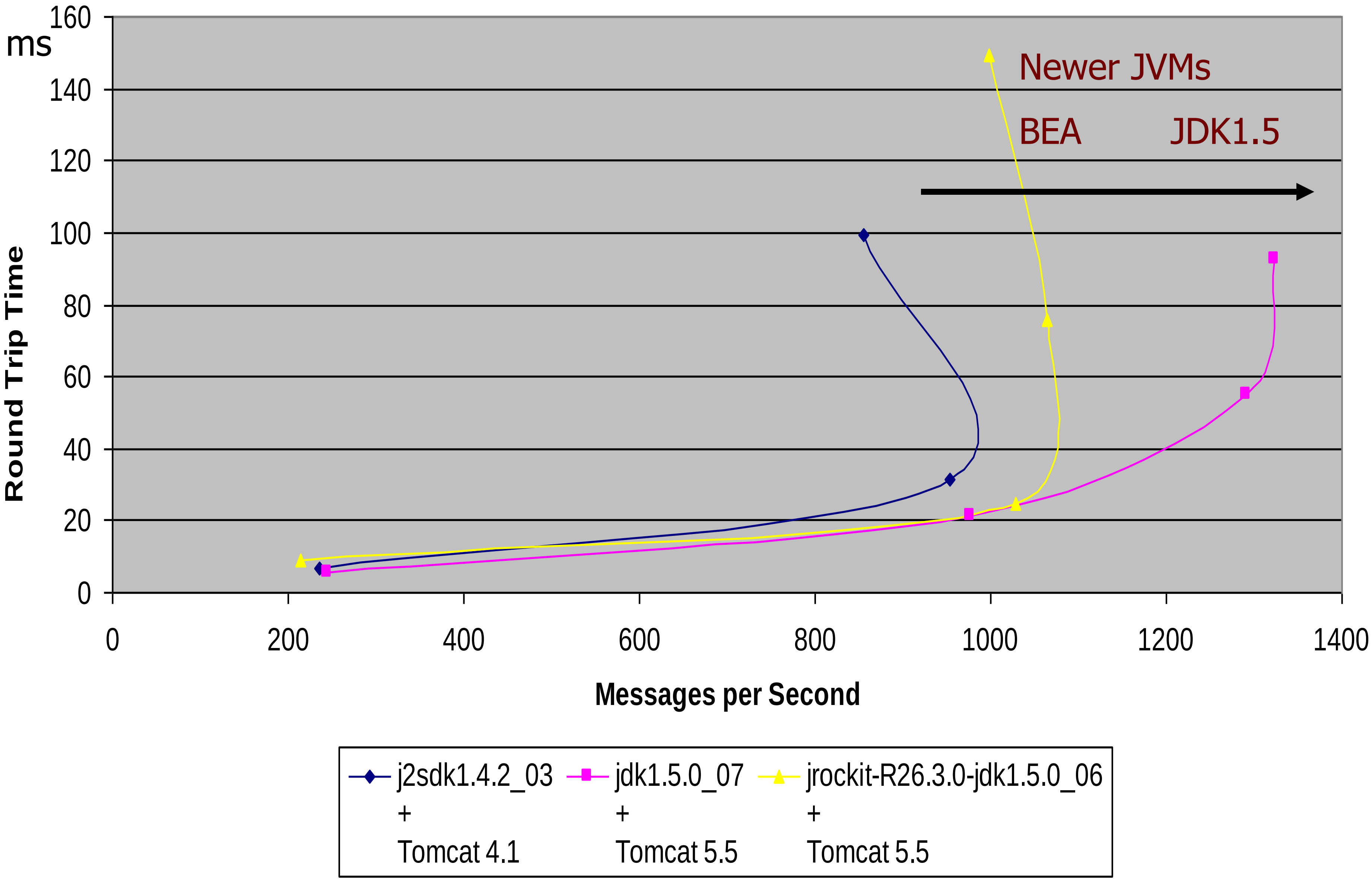
- Test III -Testing Messaging scalability:
  - WS-Management states that if a

Axis2 Performance on Multi Core Machines



Axis2 Performance on Different JVMs

On Intel Xeon with 400 byte messages



Future Work & Conclusion I

- This paper gives an overview of architecture and illustrates with a prototype.
  - Prototype focuses on using alternate means of transports to provide different QoS (Quality of Service) when certain transports are blocked due to network policies, presence of firewalls or due to NAT devices.
- Work is underway to demonstrate fault-tolerance of cmanagement omponents themselves (managers, messaging nodes, registry) and how it affects the overall management
- We will switch to MySQL/Javaspace implementation of WS-Context for registry – note this is compatible with UDDI
  - http://grids.ucs.indiana.edu/ptliupages/publications/SKG06-Aktas.pdf

Future Work & Conclusion II

- The scheme provides a Web Service management interface for easy configuration and deployment of middleware components with both general and application specific features
- The costs obtained are one-time initialization costs and hence acceptable.
- We provided basic tests for scalability purposes and are currently investigating solutions that would improve the scalability in heterogeneous and wide-area (cross-continent) resource management.
- Might be useful for debugging framework as detects and reports errors