

# Framework for High Performance Grid and Web Services

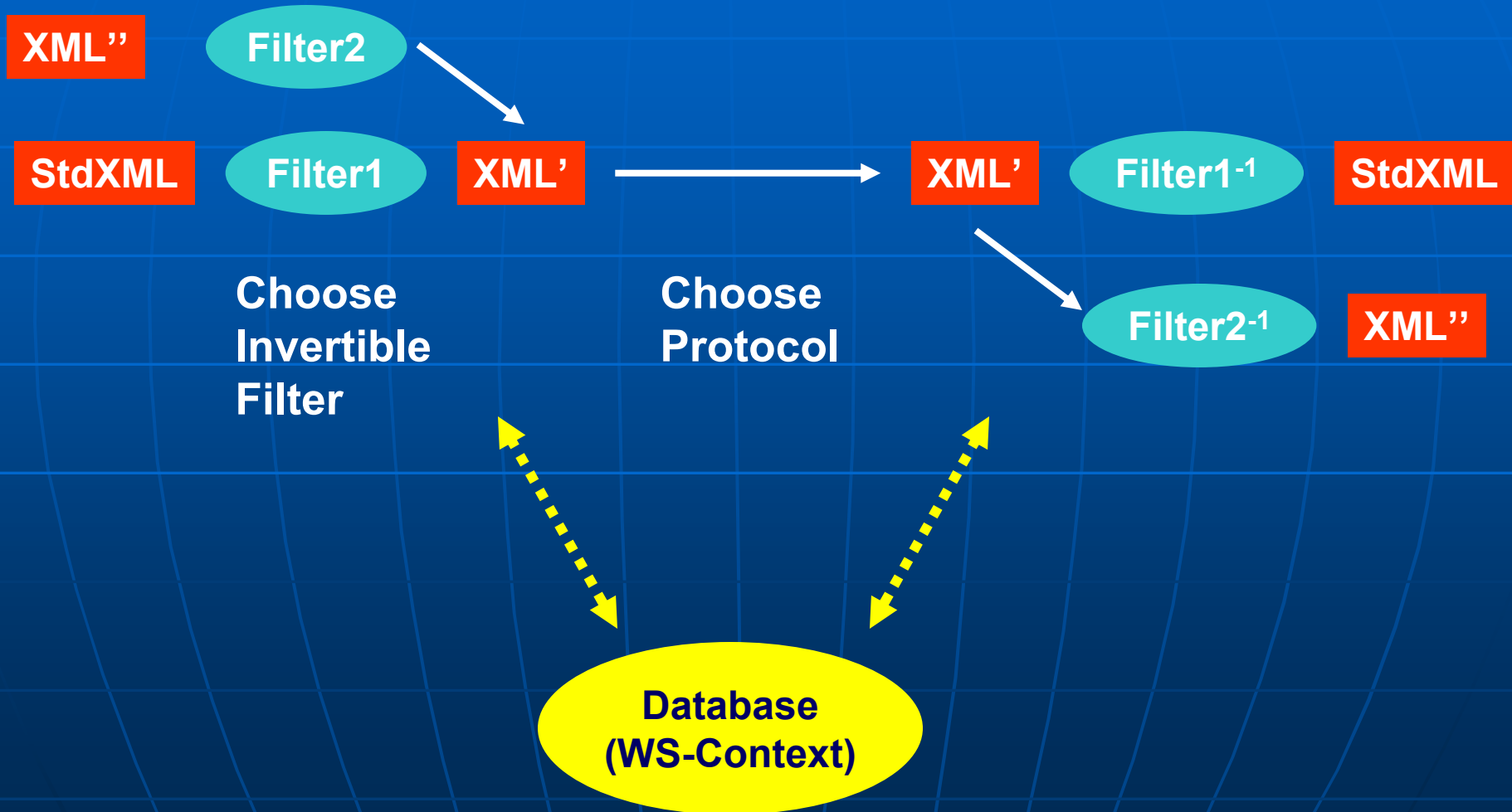
**GGF15 October 2 2005**

**Geoffrey Fox**

**Computer Science, Informatics, Physics  
Pervasive Technology Laboratories  
Indiana University Bloomington IN 47401**

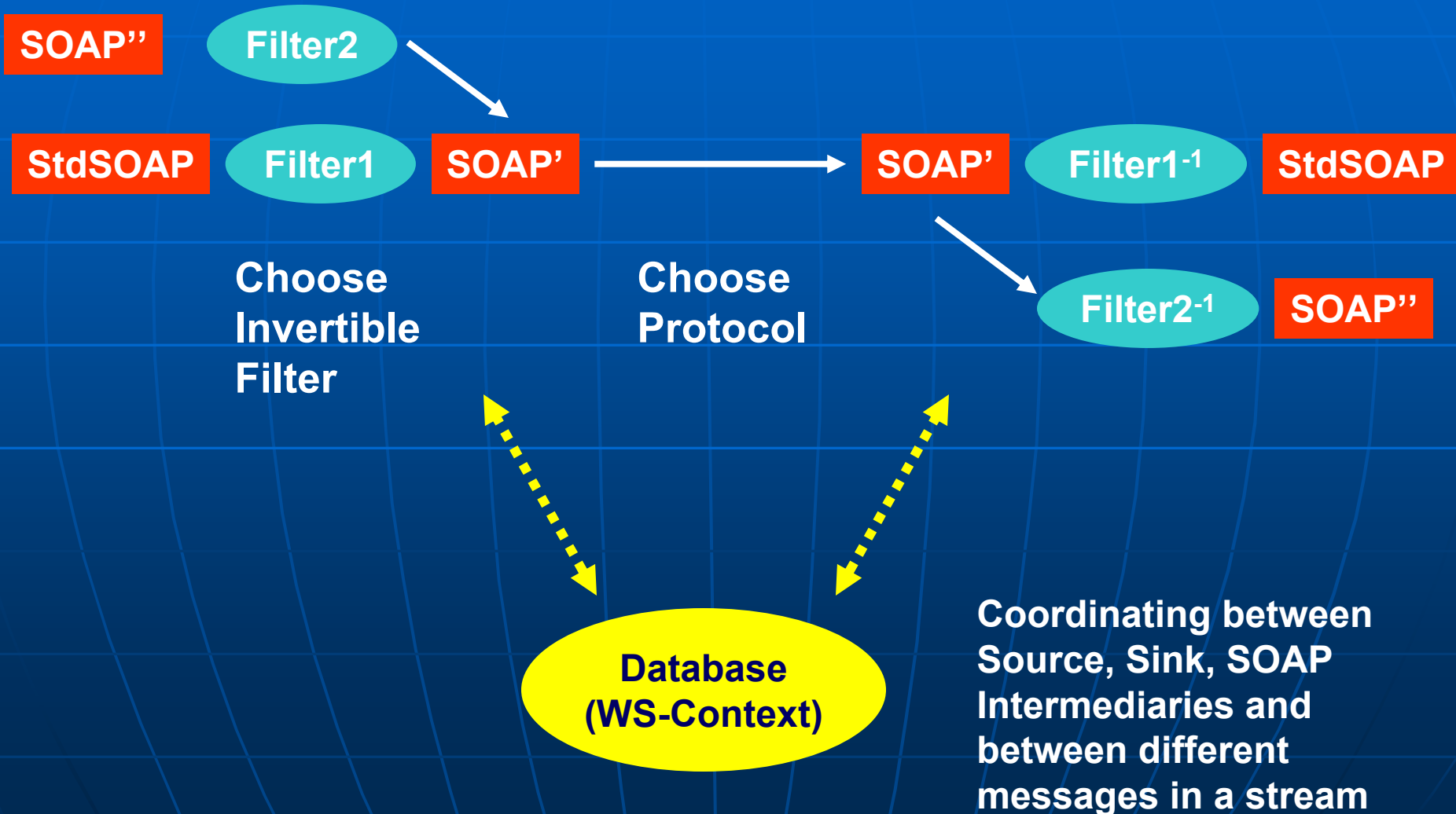
# Support for Messages

- Optimize XML representation and transport protocol



# Support for Streams

- Optimize Stream representation and transport protocol



# Basic Idea

- Bind SOAP to message-oriented middleware that virtualizes representation and protocol
- Build handlers that convert representation and support different protocols
- Specify methodology and specific parameters as meta-data stored in a meta-data (context) service
- Combination of message and context is self-describing
  - Message has a URI referencing context
- Target Axis2 as it supports SOAP Infoset and has a reasonable handler model
  - Some complicated issues here – initially porting technology to Axis2 – might do as Apache project
- Initial targets sensors, mobile devices and audio-video conferencing

# Metadata and Service Context

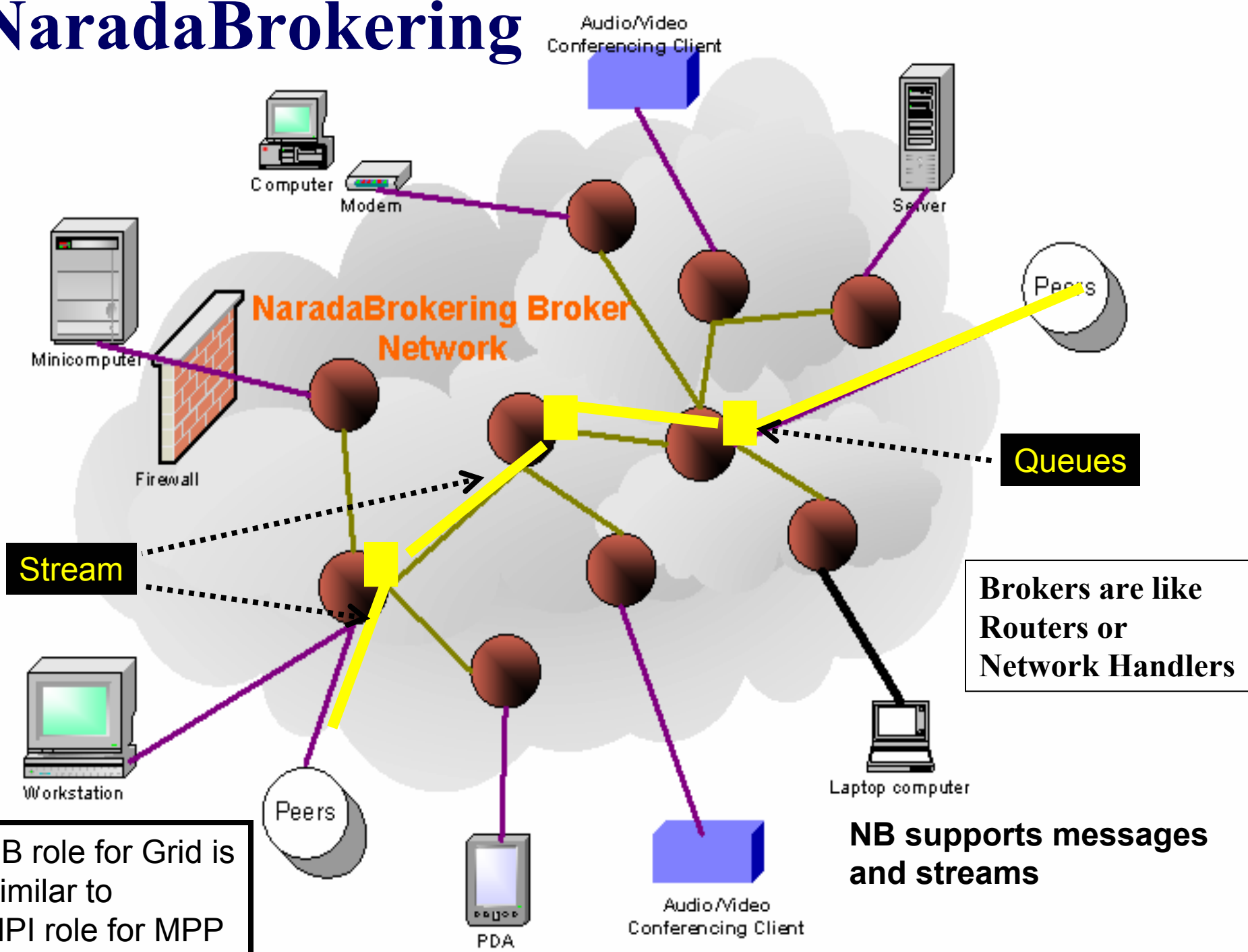
- Consider a collection of services working together
  - Workflow tells you how to specify service interaction but more basically there is shared information or context specifying/controlling collection
- WS-RF and WS-GAF have different approaches to contextualization – supplying a common “context” which at its simplest is a token to represent state
- More generally core shared information includes dynamic service metadata and the equivalent of configuration information.
- One can support such a common context either as pool of messages or as message-based access to a “database” (Context Service)
- Two services linked by a stream are perhaps simplest example of a collection of services needing context
- Note that there is a tension between storing metadata in **messages** and **services**.
  - **This is shared versus distributed memory debate in parallel computing**

# Role of WS-Context

- There are many WS-\* specifications addressing **meta-data** and both many approaches and many trade-offs
- There are **Distributed Hash Table** based systems (Chord) to achieve scalability in large scale networks
- **Managed dynamic workflows** as in sensor integration and collaboration require
  - **Fault-tolerance** and ability to support **dynamic changes** with **few millisecond** delay
  - But only a **modest number** of involved services (up to 1000's in a **session**)
  - Need **Session** NOT Service/Resource meta-data so don't use WS-RF
- We have built a **WS-Context** compliant metadata catalog supporting distributed or central paradigms
- Use for **OGC Web catalog** service with **UDDI** for slowly varying meta-data
- **3 XML Databases: UDDI WS-Context WFS** stored as **SQL**



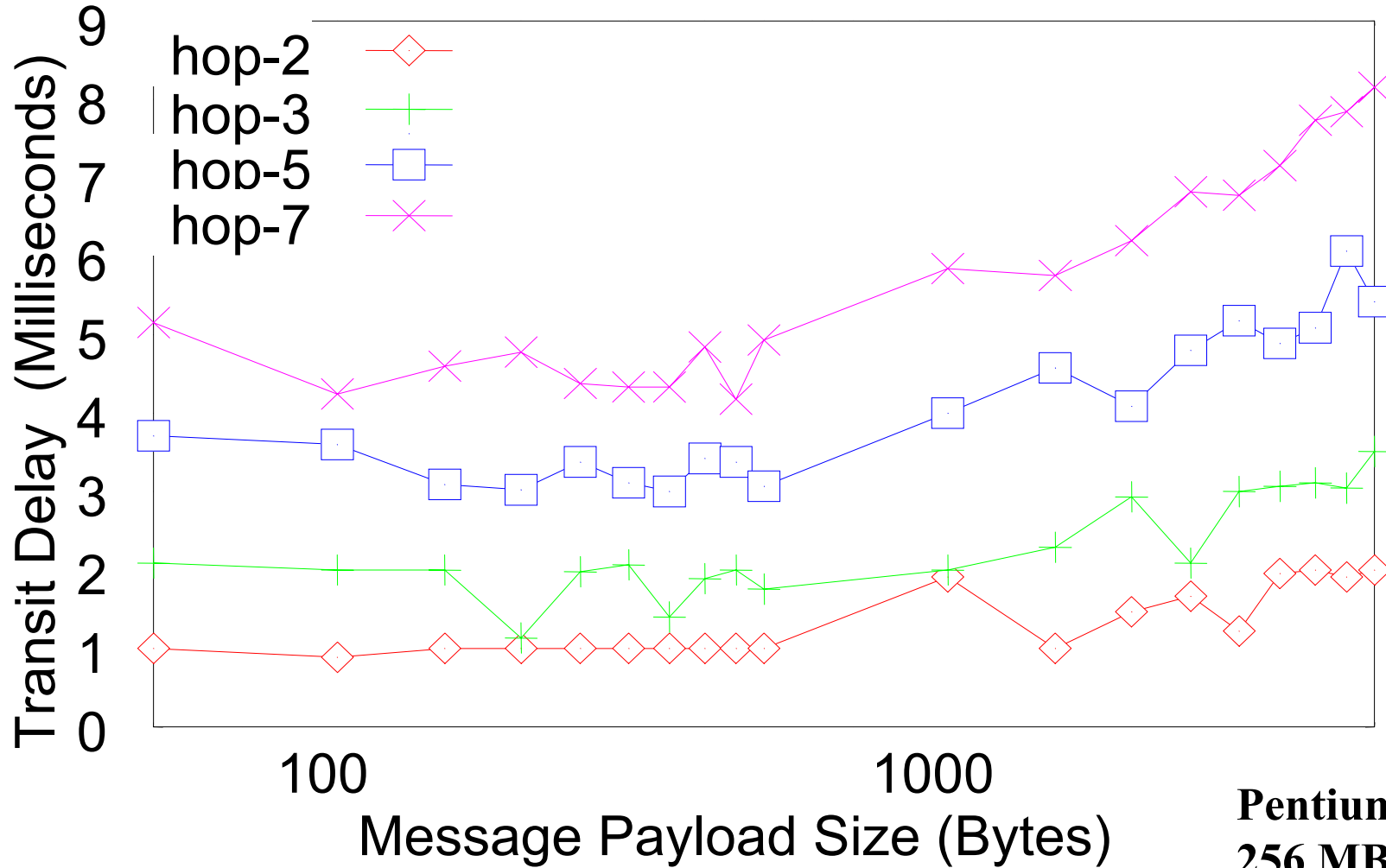
# NaradaBrokering



# Traditional NaradaBrokering Features

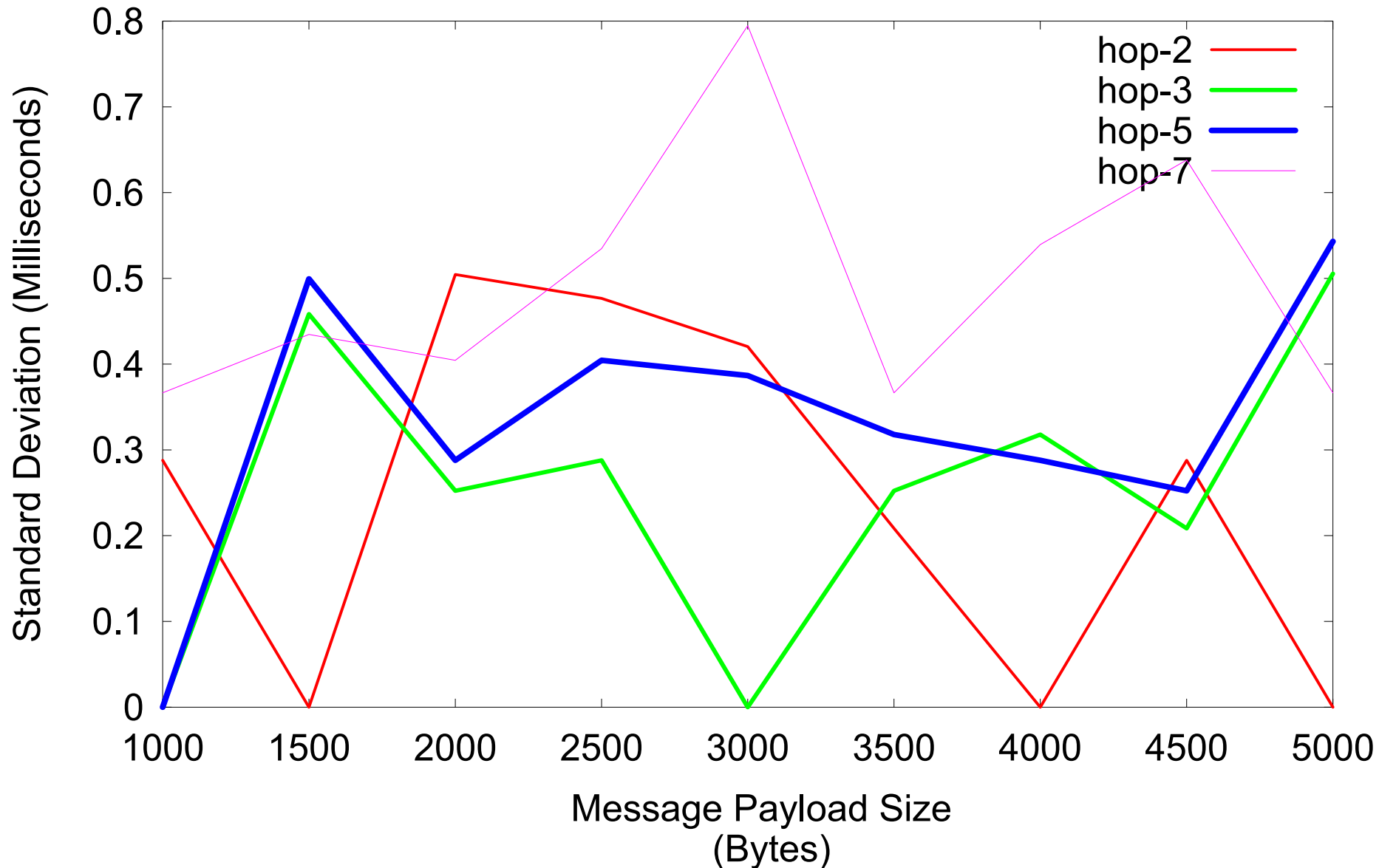
<b>Multiple protocol transport support</b> In publish-subscribe Paradigm with different Protocols on each link	Transport protocols supported include TCP, Parallel TCP streams, UDP, Multicast, SSL, HTTP and HTTPS. Communications through authenticating proxies/firewalls & NATs. Network QoS based Routing Allows Highest performance transport
<b>Subscription Formats</b>	Subscription can be Strings, Integers, <b>XPath</b> queries, <b>Regular Expressions</b> , <b>SQL</b> and tag=value pairs.
<b>Reliable delivery</b>	<b>Robust</b> and <b>exactly-once delivery</b> in presence of failures
<b>Ordered delivery</b>	<b>Producer Order</b> and <b>Total Order</b> over a message type. <b>Time Ordered</b> delivery using Grid-wide <b>NTP based absolute time</b>
<b>Recovery and Replay</b>	<b>Recovery from failures</b> and disconnects. <b>Replay</b> of events/messages at any time. <b>Buffering</b> services.
<b>Security</b>	<b>Message-level WS-Security</b> compatible security
<b>Message Payload</b> options	<b>Compression</b> and <b>Decompression</b> of payloads <b>Fragmentation</b> and <b>Coalescing</b> of payloads
<b>Messaging Related Compliance</b>	Java Message Service ( <b>JMS</b> ) 1.0.2b compliant Support for routing P2P <b>JXTA</b> interactions.
<b>Grid Feature Support</b>	NaradaBrokering enhanced <b>Grid-FTP</b> . Bridge to <b>Globus GT3</b> .
<b>Web Services supported</b>	Implementations of <b>WS-ReliableMessaging</b> , <b>WS-Reliability</b> and <b>WS-Eventing</b> .

# 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



# NB Features Released 2005-2006

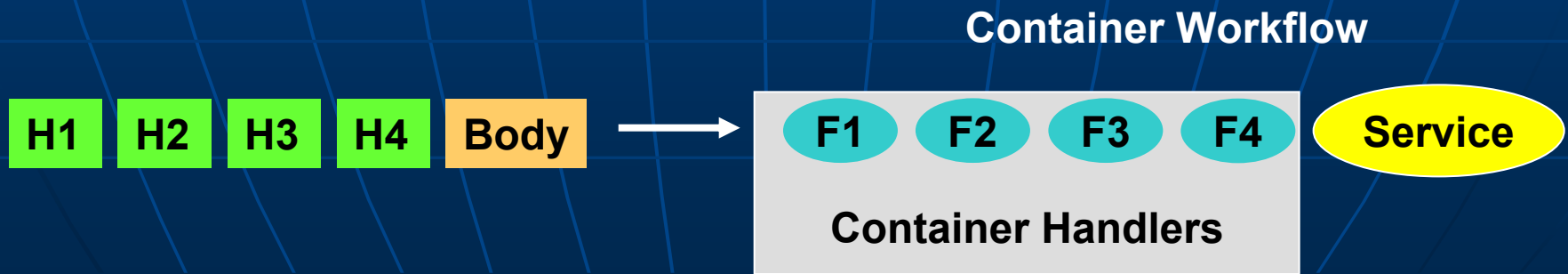
- Production implementations of **WS-Eventing**, **WS-RM** and **WS-Reliability**.
  - **WS-Notification** when specification agreed
- **SOAP message support** and NaradaBrokers viewed as SOAP Intermediaries
- **Active replay** support: Pause and Replay live streams.
- **Stream Linkage**: can link permanently multiple streams – using in annotating real-time video streams
- **Replicated storage support** for fault tolerance and resiliency to storage failures.
- **Management**: HPSearch Scripting Interface to administer streams and brokers (uses **WS-Management**)
- **Broker Topics and Message Discovery**: Locate appropriate
- Support of IBM **MQSeries** functionality and Legacy MQSeries Systems as a Grid of Grids gateway

# Possible NaradaBrokering Futures

- Clarification and expansion of **NB Broker** to act as a **WS container**
- Integration with **Axis 2.0** as **Message Oriented Middleware infrastructure**
- Support for **High Performance transport and representation for Web Services**
- **Performance based routing**
  - The broker network will dynamically respond to changes in the network based on metrics gathered at individual broker nodes.
- **Replicated publishers** for fault tolerance
- **Pure client P2P** implementation (originally we linked to JXTA)
- **Security Enhancements** for fine-grain topic authorization, multi-cast keys, Broker attacks

# SOAP Message Structure I

- SOAP Message consists of **headers and a body**
  - Headers could be for Addressing, WSRM, Security, Eventing etc.
- **Headers** are processed by **handlers or filters** controlled by container as message enters or leaves a service
- **Body** processed by Service itself
- The header processing defines the “**Web Service Distributed Operating System**”
- **Containers** queue messages; control processing of headers and offer convenient (for particular languages) service interfaces
- **Handlers are really the core Operating system services** as they receive and give back messages like services; they just process and perhaps modify different elements of SOAP Message – WS standards specify handler structure



# SOAP Message Structure II

- Content of individual headers and the body is defined by XML Schema associated with WS-\* headers and the service WSDL
- **SOAP Infoset** captures header and body structure
- **XML Infoset** for individual headers and the body capture the details of each message part
- Web Service Architecture requires that we **capture Infoset structure** but does **not** require that we represent XML in **angle bracket <content>value</content>** notation



# High Performance XML I

- There are many approaches to efficient “binary” representations of **XML Infosets**
  - MTOM, XOP, Attachments, Fast Web Services
  - DFDL is one approach to specifying a binary format
- Assume URI-S labels **Scheme** and URI-R labels **realization of Scheme** for a particular message i.e. URI-R defines specific layout of information in each message
  - DFDL from GGF quite interesting for this
- Assume we are interested in conversations where a **stream of messages** is exchanged between two services or between a client and a service i.e. two end-points
- Assume that we need to **communicate fast** between end-points that understand scheme URI-S but must **support conventional representation** if one end-point does not understand URI-S

# High Performance XML II

- First Handler **Ft=F1** handles **Transport protocol**; it negotiates with other end-point to establish a **transport conversation** which uses either HTTP (default) or a different transport such as UDP with WSRM implementing reliability
  - **URI-T** specifies transport choice
- Second Handler **Fr=F2** handles **representation** and it negotiates a **representation conversation** with scheme **URI-S** and realization **URI-R**
  - **Negotiation identifies parts of SOAP header that are present in all messages in a stream and are ONLY transmitted ONCE**
- **Fr** needs to negotiate with Service and other handlers illustrated by **F3** and **F4** below to decide what representation they will process

F1

F2

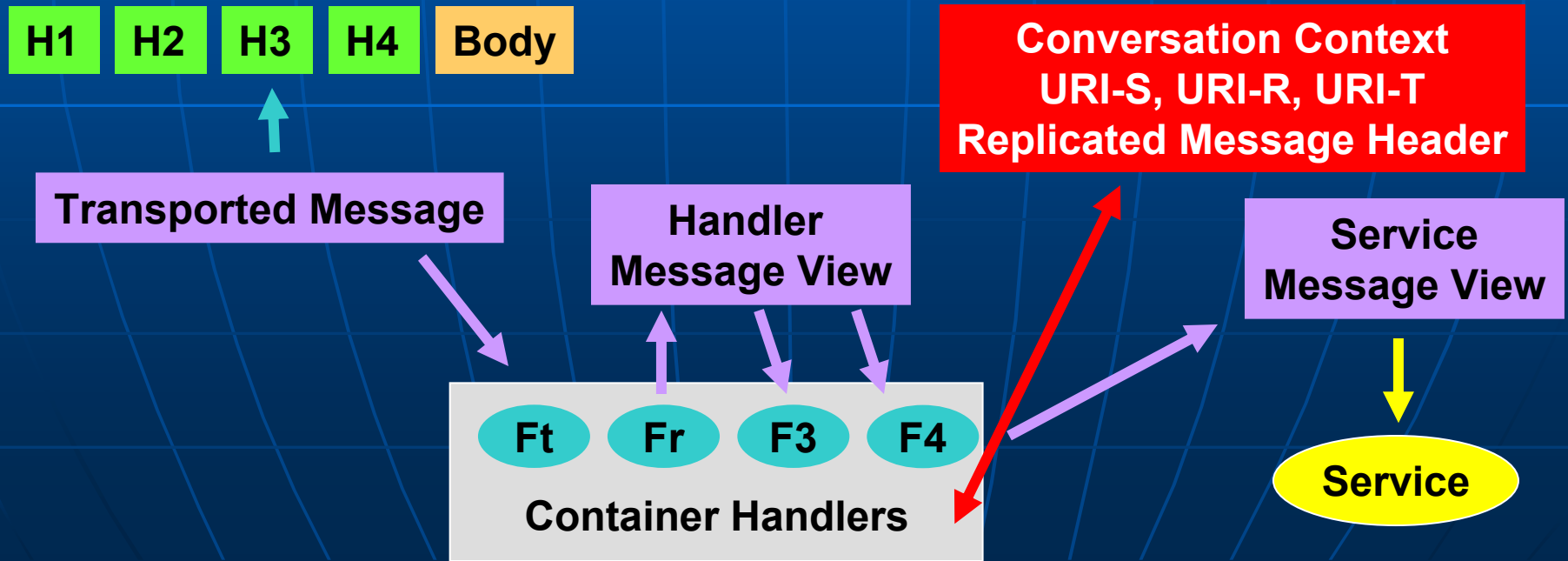
F3

F4

Container Handlers

# High Performance XML III

- Filters controlled by Conversation Context **convert messages between representations** using permanent context (metadata) catalog to hold conversation context
- Different **message views** for each end point or even for individual handlers and service within one end point
  - **Conversation Context** is fast dynamic metadata service to enable conversions
- **NaradaBrokering** will implement **Fr** and **Ft** using its support of multiple transports, fast filters and message queuing;



# Location of software for Grid Projects in Community Grids Laboratory

- <http://www.naradabrokering.org> provides Web service (and JMS) compliant **distributed publish-subscribe messaging** (software overlay network)
- <http://www.globlmmcs.org> is a **service oriented (Grid) collaboration environment** (audio-video conferencing)
- <http://www.crisisgrid.org> is an OGC (open geospatial consortium) Geographical Information System (GIS) compliant **GIS and Sensor Grid**
- <http://www.opengrids.org> has WS-Context, Extended UDDI etc.
- The work is still in progress but NaradaBrokering is quite mature
- **All software is open source** and freely available