Web Service Grids

Grid Summer School

July 22 2004

Geoffrey Fox

Community Grids Lab

Indiana University

gcf@indiana.edu

Acknowledgements

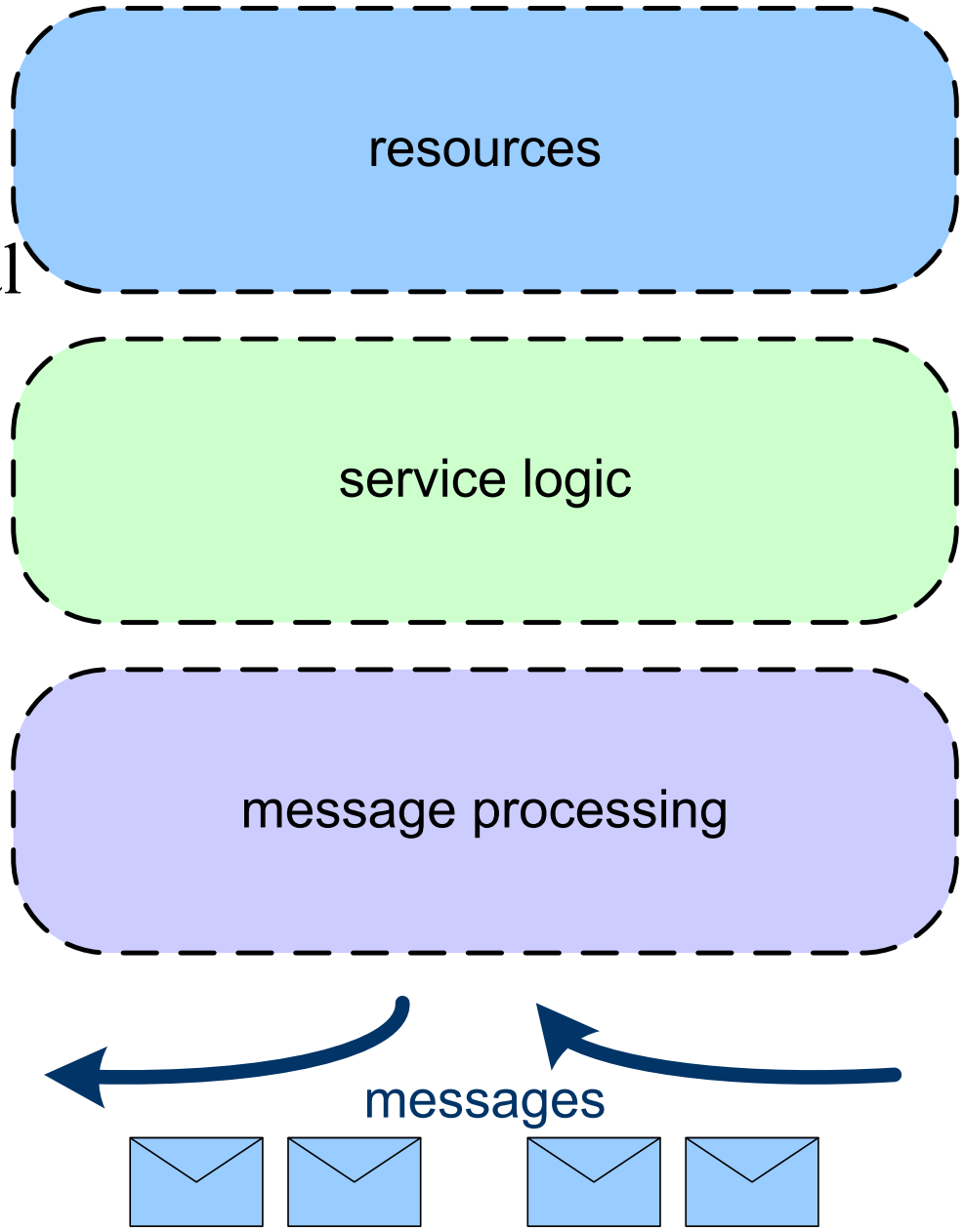
- The point of view and discussion of service architectures described here largely comes from unpublished paper by Malcolm Atkinson, David DeRoure, Alistair Dunlop, Geoffrey Fox, Peter Henderson, Tony Hey, Norman Paton, Steven Newhouse, Savas Parastatidis, Anne Trefethen and Paul Watson
- http://grids.ucs.indiana.edu/ptliupages/publications/WebServiceGrids.pdf
- The particular presentation and any mistakes are responsibility of Fox

Philosophy of Web Service Grids

- Much of distributed Computing was built by natural extensions of computing models developed for sequential machines
- This leads to the distributed object (DO) model represented by Java and CORBA
  - RPC (Remote Procedure Call) or RMI (Remote Method Invocation) for Java
- Key people think this is not a good idea as it scales badly and ties distributed entities together too tightly
  - Distributed Objects Replaced by Services
- Note CORBA was too complicated in both organization and proposed infrastructure
  - and Java was considered as "tightly coupled to Sun"
  - So there were other reasons to discard
- Thus replace distributed objects by services connected by "one-way" messages and not by request-response messages

Service Oriented Architectures I

- A service is the logical (electronic) manifestation of some physical or logical resources (like databases, programs, devices, humans, etc.) and/or some application logic that is exposed to the network;
- Service interaction is facilitated by message exchanges.



Microsoft on Services

- **Microsoft:** Service orientation is a means for building distributed systems.
  - At its most abstract, service orientation views everything from the mainframe application to the printer to the shipping dock clerk to the overnight delivery company as a service provider.
  - Service providers expose capabilities through interfaces.
  - Service-oriented architecture maps these capabilities and interfaces so they can be orchestrated into processes.
  - Orchestration = Choreography = Workflow
  - The service model is "fractal": the newly formed process is a service itself, exposing a new, aggregated capability.

Service Oriented Architectures II

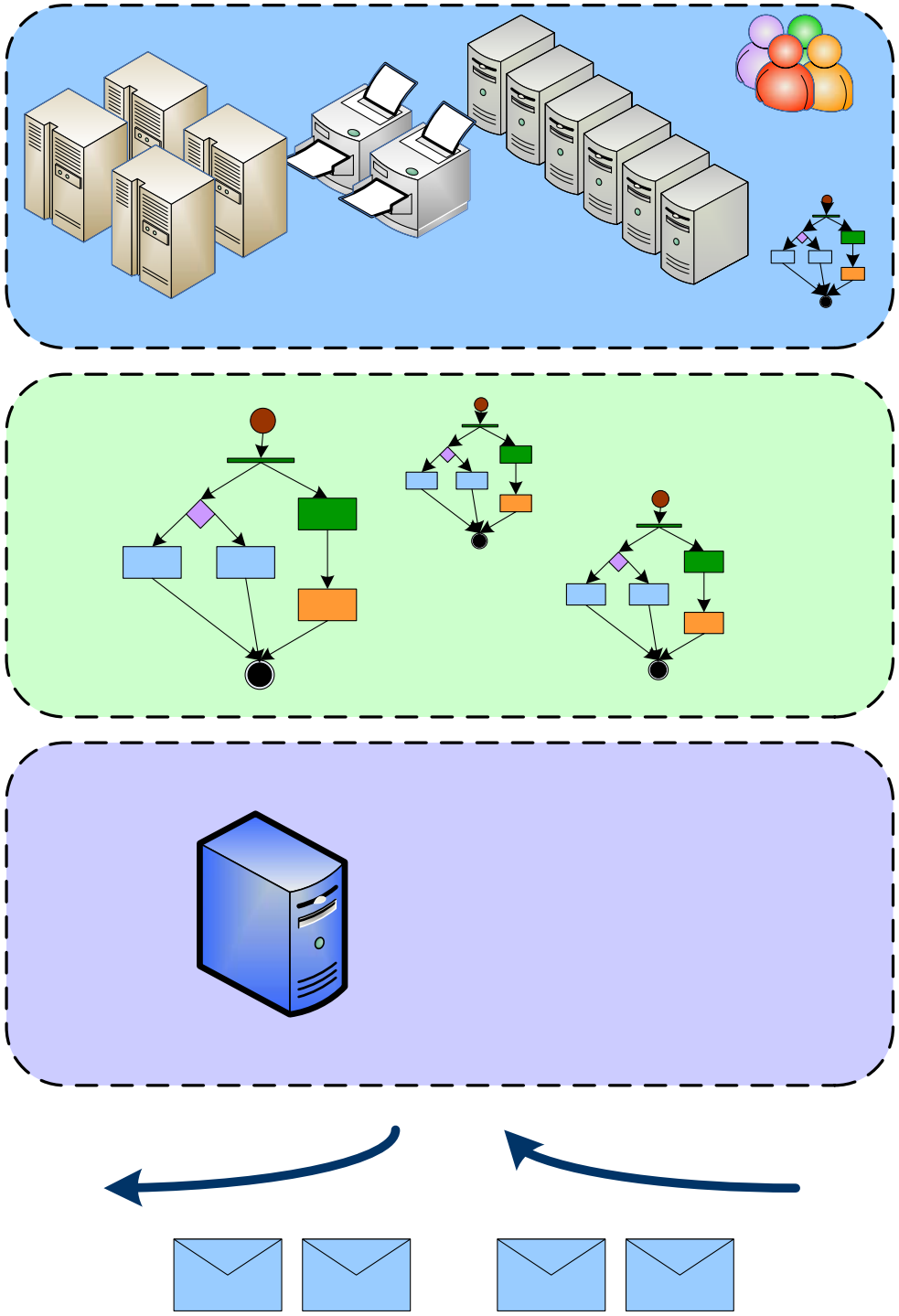
- **Service boundaries are explicit:** The boundaries of a service are well defined when they are incorporated into a distributed application. Other services do not see the internal workings, implementation details, or resource representations of a service.
- **Services are autonomous:** Service implementations are developed and evolve independently from one another.
  - NOT true of typical Java-based “systems”/”frameworks” even though recommended by software engineering principles
  - Message-based interactions encourages better design than method-based
  - Inheritance and even Java interfaces encourage spaghetti classes
- **Services can be aggregated:** Services defining their interfaces and policy can be linked together into a larger composed Web service whose detailed composition need not be exposed to other services invoking the aggregate service.

Service Oriented Architectures III

- **Services share schema and contract, not classes:** In service-oriented architectures, no single set of abstractions (classes) spans an entire application. Services share schemas (contracts) that define the structure of the information that they exchange, not information about their underlying type systems.
  - The loose-coupling assertion
- **Policies determine service compatibility:** Services interact with one another only after it has been determined – based on policy assertions – that they can meaningfully exchange information.
- Designing a **Service-oriented architecture** is the art of modeling an (virtual) organization's operational processes, as a well-factored portfolio of **network-addressable enterprise components**
- Design **Services to Last**; Design **Systems to Change**
- Separate the **interface** and the **implementation**
- Distributed Objects (e.g. Java) with a WSDL Interface are not necessarily services as defined here
  - They have a service interface

Web services

- Web Services build loosely-coupled, distributed applications, based on the SOA principles.
- Web Services interact by exchanging messages in SOAP format
- The contracts for the message exchanges that implement those interactions are described via WSDL interfaces.



Importance of SOAP

- SOAP defines a very obvious message structure with a header and a body
- The header contains information used by the "Internet operating system"
  - Destination, Source, Routing, Context, Sequence Number …
- The message body is only used by the application and will never be looked at by "operating system" except to encrypt, compress etc.
- Much discussion in field revolves around what is in header!
  - e.g. WSRF adds a lot to header

Consequences of Rule of the Millisecond

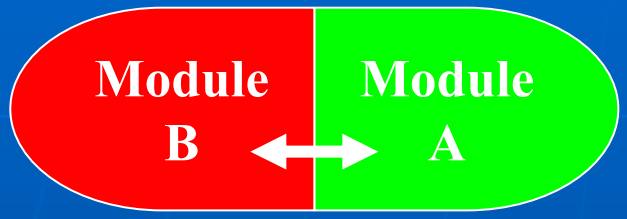
- Useful to remember critical time scales
  - 1) 0.000001 ms – CPU does a calculation
  - 2) 0.001 to 0.01 ms – MPI latency
  - 3) 1 to 10 ms – wake-up a thread or process
  - 4) 10 to 1000 ms – Internet delay
- 4) implies geographically distributed metacomputing can't compete with parallel systems
- 3) << 4) implies RPC not a critical programming abstraction as it ties distributed entities together and gains a time that is typically only 1% of inevitable network delay
  - However many service interactions are at their heart RPC but implemented differently at times e.g. asynchronously
- 2) says MPI is not relevant for a distributed environment as low latency cannot be exploited
- Even more serious than using RMI/RPC, current Object paradigms are also lead to mixed up services with unclear boundaries and autonomy
- Web Services are only interesting model for services today

What is a Simple Service?

- Take any system – it has multiple functionalities
  - We can implement each functionality as an independent distributed service
  - Or we can bundle multiple functionalities in a single service
- Whether functionality is an independent service or one of many method calls into a "glob of software", we can always make them as Web services by converting interface to WSDL
- Simple services are gotten by taking functionalities and making as small as possible subject to "rule of millisecond"
  - Distributed services incur messaging overhead of one (local) to 100's (far apart) of milliseconds to use message rather than method call
  - Use scripting or compiled integration of functionalities ONLY when require <1 millisecond interaction latency
- Apache web site has many projects that are multiple functionalities presented as (Java) globs and NOT (Java) Simple Services
  - Makes it hard to integrate sharing common security, user profile, file access .. services

Linking Modules

Closely coupled Java/Python …



Method Calls
.001 to 1 millisecond

Coarse Grain Service Model

0.1 to 1000 millisecond latency

- **From method based to RPC to message based to event-based**

What is a Grid I?

- You won't find a clear description of what is Grid and how does differ from a collection of Web Services
  - I see no essential reason that Grid Services have different requirements than Web Services
  - There may be better service-building models than that presented by Axis or .NET
  - Notice "service-building model" is like programming language – very personal!
  - Geoffrey Fox, David Walker, *e-Science Gap Analysis*, June 30 2003. Report UKeS-2003-01, http://www.nesc.ac.uk/technical_papers/UKeS-2003-01/index.html.
- Grids were once defined as "Internet Scale Distributed Computing" but this isn't good as Grids depend as much if not more on data as well as simulations

What is a Grid II?

- So Grids can be termed "Internet Scale Distributed Simple Services" and represent a way of collecting services together in same way that program (package) collects methods and objects together.
- In this view, Grids are naturally and critically tied to Web Services and so must be built on top of Web service standards
- The high performance computing and e-Science origin of Grids does give some special challenges
  - Discussed later and high bandwidth messaging is one of most serious challenges
- Grids are built with Web Services and so a Grid Service is a Web Service and differences between Grid and Web services are not important for many Grid applications
- We will explain the WS-I+ Web Service approach to Grids

Build the Internet on the Internet

- The messaging and other Web Service standards are essentially building a new Internet protocol using a software overlay network at application layer of OSI stack
  - We can't change current Internet easily and its too inflexible!
- SOAP header plus SOAP encoded negotiation controls the "new Internet protocols"
  - Reliability
  - Routing
  - Discovery of virtualized addresses mimicking DNS
  - Addressing including multicast
  - Response patterns (collective communication in MPI)
  - Security
  - Streaming
- Will enable better performance and better reliability with Web Service messaging
  - Opposite to normal complaint that SOAP Slow!!
  - Likely to use UDP based fast simple transports
- Important for P2P Networks as these are typically based on Software Overlay Networks and provide some of these messaging features

Web Services

- Java is very powerful partly due to its many "frameworks" that generalize libraries e.g.
  - Java Media Framework
  - Java Database Connectivity JDBC
- Web Services have a correspondingly collections of specifications that represent critical features of the distributed operating systems for "Grids of Simple Services"
  - Some 60 active WS-* specifications for areas
  - **a. Core Infrastructure Specifications**
  - **b. Service Discovery**
  - **c. Security**
  - **d. Messaging**
  - **e. Notification**
  - **f. Workflow and Coordination**
  - **g. Characteristics**
  - **h. Metadata and State**

Core Web Service Architecture

- **XSD** XML Schema (W3C Recommendation) V1.0 February 1998, V1.1 February 2004 http://www.w3.org/XML/Schema
- **WSDL 1.1** Web Services Description Language Version 1.1, (W3C note) March 2001 http://www.w3.org/TR/wsdl) endorsed in WS-I Basic Profile 1.0 April 2004 http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html
- **WSDL 2.0** Web Services Description Language Version 2.0, (W3C under development) March 2004 http://www.w3.org/2002/ws/desc/
- **SOAP 1.1** (W3C Note) V1.1 Note May 2000, V1.2 Recommendation June 2003 http://www.w3.org/TR/soap/, V1.1 endorsed in WS-I Basic Profile 1.0
- **SOAP 1.2** (W3C Recommendation) June 24 2003 http://www.w3.org/TR/soap/

Web Service Registry/Discovery I

- **UDDI** (Broadly Supported OASIS Standard) V3 August 2003 http://www.uddi.org/
  - UDDI is a well established OASIS service discovery standard
- **WS-Discovery** Web services Dynamic Discovery (Microsoft, BEA, Intel …) February 2004 http://ftpna2.bea.com/pub/downloads/ws-discovery.pdf
  - Addresses dynamic discovery but reliance on hardware multi-cast a limitation
- **WS-IL** Web Services Inspection Language, (IBM, Microsoft) November 2001 http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html

Web Service Registry/Discovery II

- UDDI known as suitable for relatively static applications with a peculiat construct tModel for storing information
- UDDI has limitations as to what is stored, how dynamically can be changed and nature of queries
- Maybe problems due to implementations and not standard
- It is naturally supported by a database of service locations and a description of their use using tModel flexibility
  - So should be able to extend queries, semantic richness
- **Discovery will be called "UDDI"** even if very different as UDDI blessed by WS-I
- Combining ideas from UDDI, WS-Discovery and P2P Networks seems promising

Web Service Security I

- **SAML** Security Assertion Markup Language (OASIS) V1.1 May 2004 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
- **XACML** eXtensible Access Control Markup Language (OASIS) V1.0 February 2003 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
- **WS-Security 2004** Web Services Security: SOAP Message Security (OASIS) Standard March 2004 http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
- with WS-I Basic Security Profile May 12 2004 http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2004-05-12.html

Web Service Security II

- **WS-SecurityPolicy** Web Services Security Policy (IBM, Microsoft, RSA, Verisign) Draft December 2002 http://www-106.ibm.com/developerworks/library/ws-secpol/ (WS-Security+WS-Policy)
- **WS-Trust** Web Services Trust Language (BEA, IBM, Microsoft, RSA, Verisign …) May 2004 http://www-106.ibm.com/developerworks/webservices/library/specification/ws-trust/
- **WS-SecureConversation** Web Services Secure Conversation Language (BEA, IBM, Microsoft, RSA, Verisign …) May 2004
- http://www-106.ibm.com/developerworks/library/specification/ws-secon/
- **This "builds overlay network equivalent" of SSL/HTTPS**
- **WS-Federation** Web Services Federation Language (BEA, IBM, Microsoft, RSA, Verisign) July 2003
- http://www-106.ibm.com/developerworks/webservices/library/ws-fed/

Web Service Security III

- Security is "hardest" Web Service/Grid problem and it is not clear even if there is a viable approach to some of some challenging problems such simultaneous login to multiple "dangerous resources" (supercomputers
- **WS-Security** presents the overall framework
- **WS-SecurityPolicy** defining how WS-Policy should be used to define system policy.
- **WS-Trust** is used to get authentication credentials with a Security Token Service and for example supports both PKI and Kerberos style systems.
- Often one needs to create a secure stream consisting of multiple exchanged messages; here **WS-SecureConversation** allows one to negotiate the stream security with for example a common symmetric secret key for efficient coding.
- Federation is a critical part of security solutions to both link multiple administrative domains and to efficiently support multiple resources. **WS-Federation** supports this for both security and privacy (anonymity) issues.
- **SAML** and the less well known access control markup **XACML** provide the XML schema to support Web Service security.

WS-I Interoperability

- Critical underpinning of Grids and Web Services is the gradually growing set of specifications in the Web Service Interoperability Profiles
- Web Services Interoperability (WS-I) Interoperability Profile 1.0a." http://www.ws-i.org. gives us XSD, WSDL1.1, SOAP1.1, UDDI in basic profile and parts of WS-Security in their first security profile.
- We imagine the “60 Specifications” being checked out and evolved in the cauldron of the real world and occasionally best practice identifies a new specification to be added to WS-I

Differences: WSDL and SOAP

- In WSDL 1.1, the major components were types, messages, portTypes, bindings, ports and services
- In WSDL 2.0, we have types, interfaces, bindings, endpoints and services
  - portTypes are replaced by interfaces
  - Ports are replaced by endpoints
  - Interfaces support inheritance and
  - messages are implemented with types “grouping element”
  - Operator overloading is removed
- SOAP 1.2 is pretty similar to SOAP 1.1 to the naïve reviewer

Web Service Messaging I

- **WS-Addressing** Web Services Addressing (BEA, IBM, Microsoft) March 2004 http://www-106.ibm.com/developerworks/library/specification/ws-add/
- **WS-MessageDelivery** Web Services Message Delivery (W3C Submission by Oracle, Sun ..) April 2004 http://www.w3.org/Submission/2004/SUBM-ws-messagedelivery-20040426/
- **WS-Routing** Web Services Routing Protocol (Microsoft) http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-routing.asp
- **WS-RM** Web Services Reliable Messaging (BEA, IBM, Microsoft, Tibco) v0.992 March 2004 http://www-106.ibm.com/developerworks/webservices/library/ws-rm/
- **WS-Reliability** Web Services Reliable Messaging (OASIS Web Services Reliable Messaging TC) March 2004 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm
- **SOAP MOTM** SOAP Message Transmission Optimization Mechanism (W3C) June 2004 http://www.w3.org/TR/2004/WD-soap12-mtom-20040608/

Web Service Messaging II

- **WS-Addressing** virtualizes addressing and is used in several other specifications including WSRF. It allows "end-point references" to be defined independently of the transport protocol
- **WS-MessageDelivery** is a richer specification than WS-Addressing with the interesting concept of "abstract message delivery properties" defined in a broad context including non-SOAP transport.
- **WS-RM** and **WS-Reliability** are almost identical and use message sequencing and acknowledgements to ensure guaranteed delivery of messages delivered in streams.
  - Not obviously correct for PDA's where ACK's expensive
  - Enable UDP transport with application level TCP-like retransmission
- **SOAP MOTM** defines optimized encoding for SOAP messages and partially addresses the critical need in e-Science for high performance transport
  - I think there are more powerful approaches to High Performance transport

WS-Addressing

- This expands addressing over that supported in SOAP and WSDL
- Generalized address URI to an "Endpoint Reference" containing
  - Address as any URI
  - Properties --
  - Selected portType in WSDL
  - Service-port in WSDL
  - Policy written in WS-Policy
- Message Information Header
  - Destination: URI
  - Source: Endpoint defining source of message
  - Reply: Endpoint to send replies to
  - Fault: Endpoint for faults
  - Action: Undefined URI defining semantics of message
  - MessageID: Label of message as a URI
  - Relationship: Describes URI of related message and specifies nature of relationship
  - Reply: Specifies that this is a reply to a message of a given MessageID

WS-Addressing

- Note the address is a URI (Universal Resource Identifier) that is typically a URL
- This address is then part of SOAP header and processed by SOAP Handler
- Address could be "virtual" (e.g. a topic in a publish-subscribe system) as long as SOAP handler understands this and knows how to route it
  - Bind transport system to WebSphereMQ, JMS or NaradaBrokering
- The endpoint properties in SOAP header can be used as in WSRF to enrich address and control processing
- This is yet another source of metadata

Mechanisms for Reliable Messaging I

- There are essentially sequence numbers on each message
- Unreliable transmission detected by non-arrival of a message with a particular sequence number
- Remember this is "just some TCP reliability" built at application level
- One can either use ACK's – Receiver (service B) positively acknowledges messages when received
  - Service A fully responsible for reliability
- Or NAK's – Service B is partially responsible and tracks message numbers – sends a NAK if sequence number missing

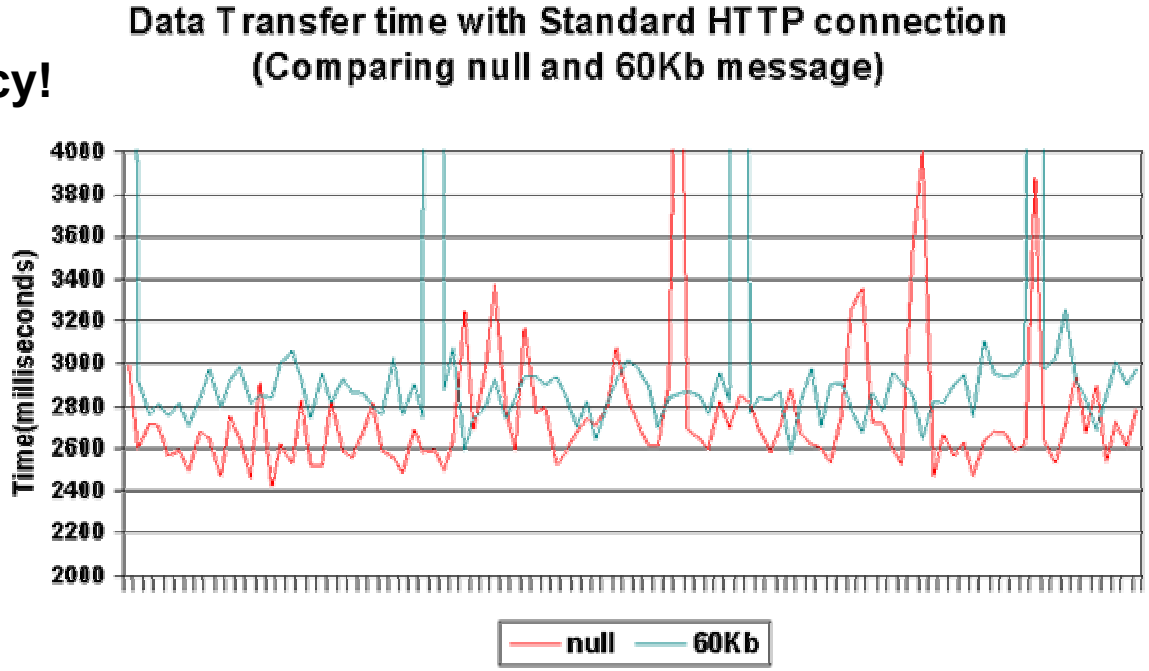
Mechanisms for Reliable Messaging II

- Each message has a retransmission time; messages are retransmitted if ACK's not received in time
  - Uses some increasing time delay if retransmit fails
- Note need to be informed (eventually) that OK to throw away messages at sender; pure NAK insufficient
- Note this is final end-point to beginning end-point: TCP reliability is for each link and has different grain size and less flexible reliability mechanisms
- There are several efficiency issues
  - Divide messages into groups and sequence within groups
  - Do not ACK each message but rather sequences of messages
- NAK based system attractive if high latency (some mobile devices) on messaging from receiver back to sender

Custom Message Reliability

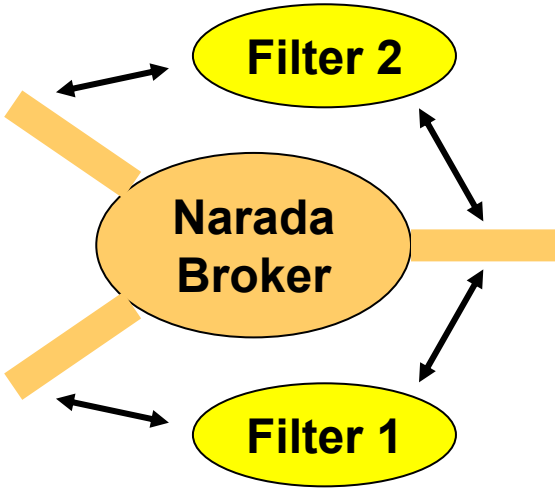
2 second PDA reply latency!

Different endpoints may well need different reliability schemes. Another reason to use application layer. NaradaBrokering offers universal support



Wireless Optimized WS-RM

WS-Reliability



Comparing some of the features in WS-Reliability and WS-ReliableMessaging I

	WS-Reliability	WS-ReliableMessaging
Related Specifications	SOAP	SOAP, WS-Addressing and WS-Policy
Acknowledgement scheme for reliable delivery	Relies only on positive acknowledgements. Error corrections are initiated by the source.	Uses both positive and negative acknowledgments. Error corrections can thus be initiated at both source and sink.
Message numbering initialization	Starts at 0 for the first message in a group.	Starts at 1 for the first message in a group.
Message number exhaustion	Sender and receiver terminate sequences if message number with Long.MAX_VALUE is received.	A `MessageNumberRollover` fault is issued by the source if message numbering exceeds Long.MAX_VALUE, and the sequence is terminated.
Message numbering information	REQUIRED only for groups with more than 1 message.	Message number is REQUIRED for every message.
Acknowledgement Ranges	Allows acknowledgement of a range of messages.	Allows acknowledgement of a range of messages.
Requesting acknowledgements	The `AckRequested` element is REQUIRED in every message for which Guaranteed delivery or Ordered delivery needs to be ensured.	`AckRequested` is used to request the receiving entity to acknowledge the message received. This is not REQUIRED for messages that are not retransmissions or the last message within a group.

Comparing some of the features in WS-Reliability and WS-ReliableMessaging I I

	WS-Reliability	WS-ReliableMessaging
Terminating group of message	Based on the agreement items of `GroupExpiryTime` `GroupMaxIdleDuration`	Based on the policy settings associated with `SequenceExpiration` and `InactivityTimeout`
Exchanges indicating group termination	No separate exchange exists for terminating a group of messages.	A specific exchange, `TerminateSequence`, exists for terminating a sequence. A source is required to issue this after getting acknowledgments on ALL messages.
Retransmissions	Triggered after the receipt of a set of positive acknowledgements.	Triggered after the receipt of a set of positive and negative acknowledgements. The `RetransmissionInterval` for a group of messages, which can be adjusted using exponential backoff algorithm also triggers it.
Quality of Service	Agreements can also be established regarding various protocol elements.	WS-Policy assertions are used to meet delivery assurances, and also to set various protocol agreements.
Delivery assurances supported	Exactly once ordered delivery, reliable delivery. Order is always tied to guaranteed delivery and cannot be separately specified.	At most once, at least once and exactly once. Order is not necessarily tied to guaranteed delivery.
Security	Relies on WS-Security and assorted specifications	Relies on WS-Security and assorted specifications
Protocol faults/error reporting	Faults issued are based on problems with message formats and message processing.	Faults issued are based on problems with message formats, message processing and message number rollovers.

Mirror Mirror on the wall
Who is the fastest most reliable of them all?
Web Services!!!

- Application layer "Internet" allows one to optimize message streams and the cost of "startup time", Web Services can deliver the fastest possible interconnections with or without reliable messaging
- Typical results from Grossman (UIC) comparing Slow SOAP over TCP with binary and UDP transport (latter gains a factor of 1000)

Record Count	Pure SOAP			SOAP over UDP			Binary over UDP		
	MB	μ	σ/μ	MB	μ	σ/μ	MB	μ	σ/μ
10000	0.93	2.04	6.45%	0.5	1.47	0.61%	0.28	1.45	0.38%
50000	4.65	8.21	1.57%	2.4	1.79	0.50%	1.4	1.63	0.27%
150000	13.9	26.4	0.30%	7.2	2.09	0.62%	4.2	1.94	0.85%
375000	34.9	75.4	0.25%	18	3.08	0.29%	10.5	2.11	1.11%
1000000	93	278	0.11%	48	3.88	1.73%	28	3.32	0.25%
5000000	465	7020	2.23%	242	8.45	6.92%	140	5.60	8.12%

SOAP Tortoise and UDP Hare II

- Mechanism only works for streams – sets of related messages
- SOAP header in streams is constant except for sequence number (Message ID), time-stamp ..
- So negotiate stream in Tortoise SOAP – ASCII XML over HTTP and TCP –
  - Deposit basic SOAP header through connection
  - Agree on firewall penetration, reliability mechanism, binary representation and fast transport protocol
  - Typically transport UDP plus WS-RM
- Fast transport (On a different port) with messages just having "FastMessagingContextToken", Sequence Number, Time stamp if needed
  - RTP packets have essentially this
  - Could add stream termination status
- Can monitor and control with original negotiation stream
- Can generate different streams optimized for different end-points

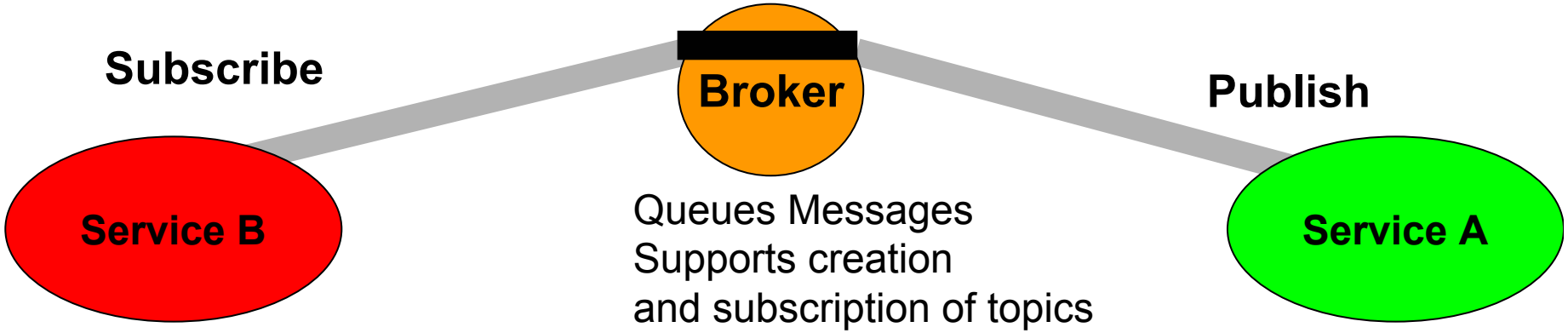
Web Service Notification I

- **WS-Eventing** Web Services Eventing (BEA, Microsoft, TIBCO) January 2004 **http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/WS-Eventing.asp**
- **WS-Notification** Framework for Web Services Notification with **WS-Topics**, **WS-BaseNotification**, and **WS-BrokeredNotification** (OASIS) OASIS Web Services Notification TC Set up March 2004 **http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn** and **http://www-106.ibm.com/developerworks/library/specification/ws-notification/**
- **JMS** Java Message Service V1.1 March 2002 **http://java.sun.com/products/jms/docs.html**

Notification Architecture

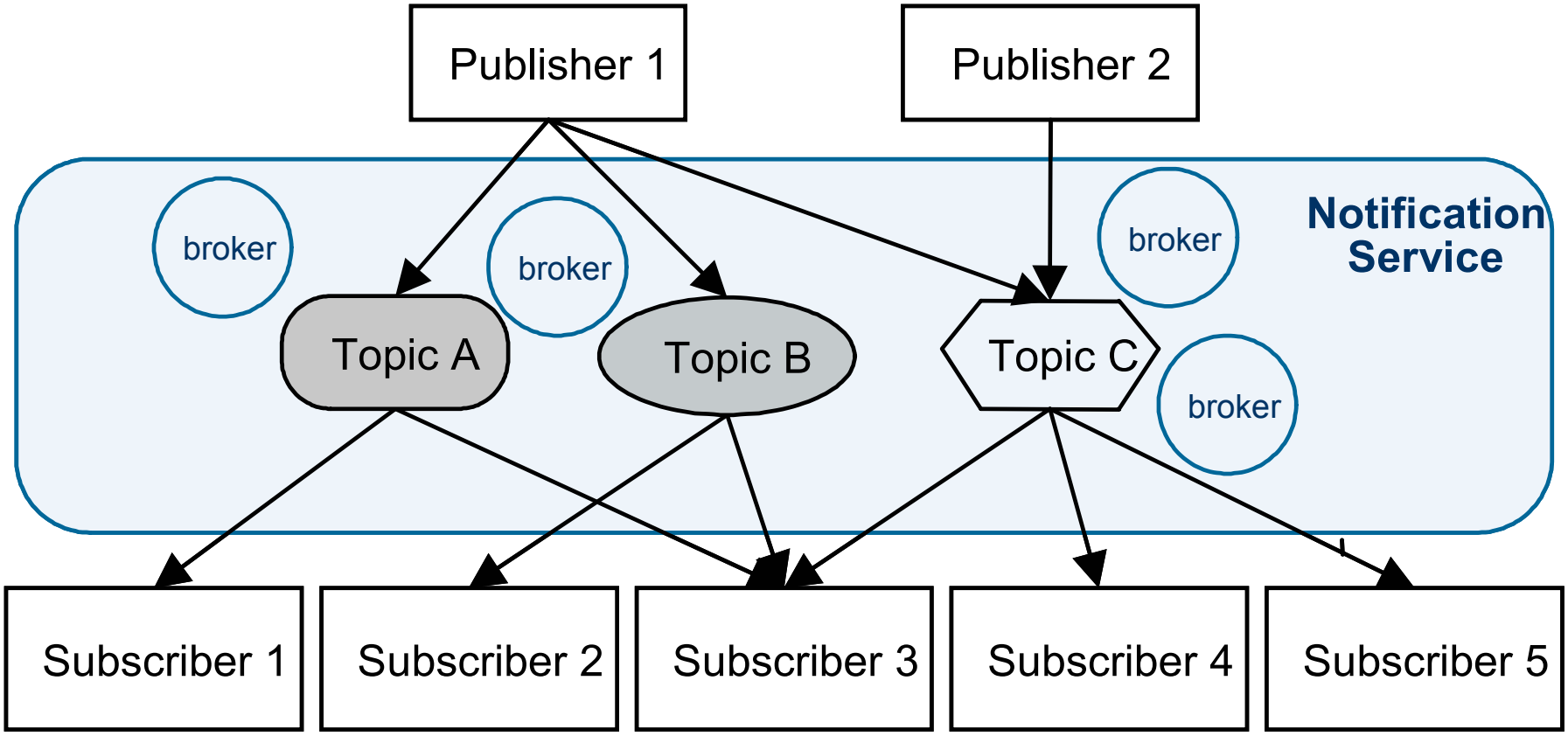
- Point-to-Point

- Or Brokered

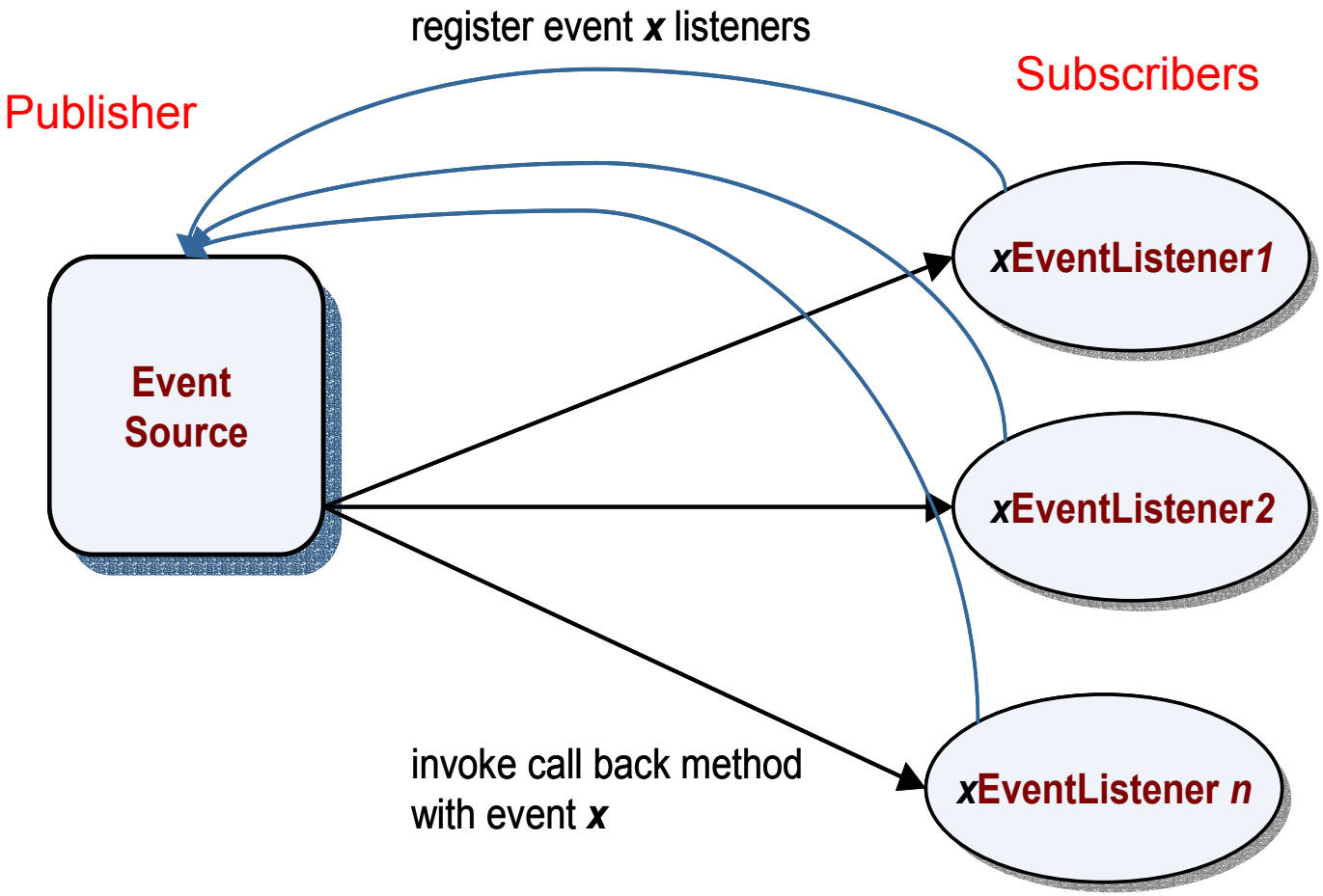


- Note that MOM (Message Oriented Middleware) uses brokered messaging for ALL transmission and not just "special" notification messages

Classic Publish-Subscribe



Event-based Programming



Java delegation event model

OS (Java VM) plays Role of broker

Web Service Notification II

- **WS-Eventing** is quite similar to WS-BaseNotification and provides service to service notification
- **WS-Notification** is similar to CORBA event service and adds brokers to mediate notification which has several advantages
  - Don't need queues and lists of subscribers on each service
  - Solution scales to any number of publishers/subscribers
- JMS well known successful non Web Service brokered notification system
- Topics defined in **WS-Topics** can also provide contextualization
- Expect this area to clarify reasonably soon

Comparison of Notification Mechanisms I

	WS-Notification	WS-BaseNotification	WS-Eventing	JMS
Related Specifications	SOAP, WS-Addressing, WS-BaseNotification, WS-Brokered Notification, WS-Topics, WS-Resource Properties and WS-ResourceLifetime	SOAP, WS-Addressing, WS-Resource Properties, WS-Topics, and WS-ResourceLifetime	SOAP, WS-Addressing	Java
Support for loosely coupled notifications. (Producers need not know consumers)	Yes. The intermediary called Notification Broker and the exchanges that need to be supported are defined in the WS-Brokered Notification specification.	No.	No.	Yes
Support for replay like features	One can get last message to a topic. A sink can also retrieve message issued between the pausing and resumption of a subscription.	One can get last message to a topic. A sink can also retrieve message issued between the pausing and resumption of a subscription.	No.	Yes. This is available for reliable subscribers through the recover option. Transient subscribers do not have this feature.
Subscription operations	Subscribe, Pause and Resume. (There is no exchange to unsubscribe).	Subscribe, Pause and Resume. (There is no exchange to unsubscribe).	Subscribe, Renew, Unsubscribe and Subscription End.	Subscribe, Unsubscribe, receive (with time constraint), recover, rollback

Comparison of Notification Mechanisms II

	WS-Notification	WS-BaseNotification	WS-Eventing	JMS
Support for filters on occurrences	YES	YES	YES	YES. The format generally specified is in SQL.
Subscription lifetimes	Defined using the WS-Resource Lifetime specification.	Defined using the WS-Resource Lifetime specification.	Contained within the Subscribe and Renew exchanges.	For persistent subscriptions a subscription is considered active till such time that an unsubscribe operation is invoked. Transient subscriptions are valid till they sign off.
Notification filters and topic expressions supported	Topic Expressions supported: QName, "/" separated Strings, and XPath path expressions.	Topic Expressions supported: QName, "/" separated Strings, and XPath path expressions.	Filter supported is XPath.	Topics are generally "/" separated strings.
Hierarchical topics and Wildcards support	Yes. Supports * and // wildcards for selection of topic descendants in a topic tree.	Topic trees could possibly be maintained in producer too. This is part of WS-Topics and WS_BaseNotification uses WS-Topics.	No.	Implementation dependant. The specification makes no specific recommendation regarding this issue.
Topic space management	Defined using WS-Topics. The topic space will also support exchanges as defined by the WS-ResourceProperties specification.	Defined using WS-Topics. The topic space will also support exchanges as defined by the WS-ResourceProperties specification.	No formal recommendation regarding topic management.	No formal recommendation regarding topic management.

Comparison of Notification Mechanisms III

	WS-Notification	WS-BaseNotification	**WS-Eventing**	**JMS**
Advertisement of supported topics	Yes. The NotificationProducer interface allows inspection of available topics.	No.	No.	No.
On demand publishing	YES. This is supported through the WS-Brokered Notification specification.	No.	No.	No.
Notification messages	Provides support for both a Notify message as well as "raw" application specific message,	Provides support for both a Notify message as well as "raw" application specific message,	Does not define any special Notification message type.	Has a well defined Message interface. This is then used to support other flavors of messages such as TextMessage, BytesMessage, ObjectMessage and StreamMessage.
Suggested Security	WS-Security and assorted specifications.	WS-Security and assorted specifications.	WS-Security & assorted specifications.	
Support for multiple delivery modes	No explicit support for reliable messaging. Defers to WSRM for this.	No explicit support for reliable messaging. Defers to WSRM for this.	No explicit support for reliable messaging. Possibly will defer to WSRM for this.	Supports PERSISTENT and NON_PERSISTET delivery modes.

CORBA Event Service

- The CORBA Event Service has more or less similar principles.
  - There is a concept of an EventChannel – similar to Broker in JMS or WS-Notification
  - There are also roles such as PushSupplier, ProxyPushConsumer, PullConsumer, and ProxyPullSupplier to facilitate push/pull operations for retrieval of events from the EventChannel.
- Either the push/pull model can be used at either end.
- The EventChannel which is a standard CORBA object is both the supplier and consumer of events, and it keeps track of suppliers and consumers through callback interfaces.
- Consumers can use either blocking/non-blocking operations for retrieval of events.

Web Services Get Together I
Coordination and Workflow, Transactions and Contextualization

- Workflow Coordination and Orchestration refer to the general integration of multiple Web Services to form another composite Service
  - Sometime called "Programming the Grid"
- Contextualization refers to providing a linkage between services clients and messages to provide a framework for stateful interactions – noite workflow can use contextualization but it is not required
- Transactions refer to important classes of workflow corresponding to classic business processes

Web Services Get Together II

- **WS-CAF** Web Services Composite Application Framework including **WS-CTX, WS-CF** and **WS-TXM** below (OASIS Web Services Composite Application Framework TC) http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-caf
- **WS-CTX** Web Services Context (OASIS Web Services Composite Application Framework TC) V1.0 July 2003 http://www.arjuna.com/library/specs/ws_caf_1-0/WS-CTX.pdf
- **WS-CF** Web Services Coordination Framework (OASIS Web Services Composite Application Framework TC) V1.0 July 2003 http://www.arjuna.com/library/specs/ws_caf_1-0/WS-CF.pdf
- **WS-TXM** Web Services Transaction Management (OASIS Web Services Composite Application Framework TC) V1.0 July 2003 http://www.arjuna.com/library/specs/ws_caf_1-0/WS-TXM.pdf

Web Services Get Together III

- **WS-Coordination** Web Services Coordination (BEA, IBM, Microsoft) September 2003 http://www-106.ibm.com/developerworks/library/ws-coor/
  - Used with **WS-AtomicTransaction** and **WS-BusinessActivity**
- **WS-AtomicTransaction** Web Services Atomic Transaction (BEA, IBM, Microsoft) September 2003 http://www-106.ibm.com/developerworks/library/ws-atomtran/
- **WS-BusinessActivity** Web Services Business Activity Framework (BEA, IBM, Microsoft) January 2004
- **BTP** Business Transaction Protocol (OASIS) May 2002 with V1.0.9.1 May 2004 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=business-transaction

Web Services Get Together IV

- **BPEL** Business Process Execution Language for Web Services (OASIS) V1.1 May 2003 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel and http://www-106.ibm.com/developerworks/library/ws-bpel/
  - Winning from importance of supporters (IBM, Microsoft)
- **WS-Choreography** (W3C) http://www.w3.org/2002/ws/chor/ V1.0 Working Draft April 2004 http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/
- **WSCL** Web Services Conversation Language (W3C Note) Submission from HP March 2002 http://www.w3.org/TR/wscl10/ not active
- None of these discusses message streams between services and so use for dataflow applications unclear

Web Services Get Together V

- **WS-Context** and **WS-Coordination** represent general approaches to contextualization.
- Three different approaches to transactions covering typical two-phase transactions as well as more complex business processes
- Each of 3 approaches packages the four component capabilities in different ways
  - Context
  - Coordination of work
  - 2 phase transaction
  - General transactions
- Not clear if workflow separate from transactions
- So an important but immature area
- The issue of "model for shared data is implicit and potentially difficult as in metadata discussion

More details on WS-Context

- Context corresponds to shared data and is roughly equivalent to a mix of Environment and Configuration variables in traditional programming
- We imagine N Web Services linked in some way
  - Maybe N=2 and linkage is message stream
  - Maybe N=2 and one Web service is a "Configuration Manager" and another a Web Service starting up
  - Maybe N=1000 and the Web Services are each controlling a cluster node
  - Maybe N=4 and we have Web services controlling CFD, Structures, Electromagnetic and optimization services
- The Context can be passed directly by putting data in message or one can indirectly specify a URI which references a Web service from which one can get the context data
- Context data is metadata defining the joint application
- Simplest example of context data is a single token allowing stateful interactions

WS-Context II

- The simplest WS-CAF concept is the shared data which is associated with an activity defined as a set of Web services
  - One can specify list
  - One can manage lifetime of context data
- The next level involves explicit coordination of the services with one or more coordination web services
  - Now we entering same regime as “workflow” but targeting specific well used simple workflows such as transactions
- It is not clear to me why coordination is not built on top of workflow languages such as BPEL
- Note XML is not terribly good at defining coordination and workflow as “control” not easy to specify
- It seems to me that shared data is important and in fact useful in workflow
  - Note that shared data could be stored in a dynamic metadata catalog with a scope defined by services in context

Web Service Characteristics

- **WS-Policy** Web Services Policy Framework (BEA, IBM, Microsoft SAP) http://www-106.ibm.com/developerworks/library/ws-polfram/
  - Used in WS-SecurityPolicy but this is not part of WS-I
  - Policy essential in negotiations that underlie many Web Service operations and seems likely WS-Policy will evolve to
- **WS-Agreement** Web Services Agreement Specification (GGF under development) http://www.gridforum.org/Meetings/GGF11/Documents/draft-ggf-graap-agreement.pdf
  - Use for specifying service level agreements

Web Service Metadata and State I

- The Semantic Grid and Semantic Web are important frameworks for metadata but handicapped by lack of "compelling" tools
- **RDF** Resource Description Framework (W3C) Set of recommendations expanded from original February 1999 standard http://www.w3.org/RDF/ and the heart of the Semantic Web and Grid http://www.semanticgrid.org
- **DAML+OIL** combining DAML (Darpa Agent Markup Language) and OIL (Ontology Inference Layer) (W3C) Note December 2001 http://www.w3.org/TR/daml+oil-reference
- **OWL** Web Ontology Language (W3C) Recommendation February 2004 http://www.w3.org/TR/2004/REC-owl-features-20040210/

Web Service Metadata and State II

- **WS-DistributedManagement** Web Services Distributed Management Framework with MUWS and MOWS below (OASIS) http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm
- Management includes issues like monitoring quality of service, enforcing service level agreements, controlling tasks and managing life-cycles.
- **WSDM-MUWS** Web Services Distributed Management: Management Using Web Services (OASIS) V0.5 Committee Draft April 2004 http://www.oasis-open.org/committees/download.php/6234/cd-wsdm-muws-0.5.pdf
- **WSDM-MOWS** Web Services Distributed Management: Management of Web Services (OASIS) V0.5 Committee Draft April 2004 http://www.oasis-open.org/committees/download.php/6255/cd-wsdm-mows-0.5-20040402.pdf
- **WS-MetadataExchange** Web Services Metadata Exchange (BEA,IBM, Microsoft, SAP) March 2004 http://www-106.ibm.com/developerworks/library/specification/ws-mex/
  - Describes how metadata can be exchanged between services rather than by looking it up in registries like UDDI or higher level metadata catalogs; the old OGSI standard used such service-resident metadata extensively

Web Service Metadata and State III

- **WS-RF** Web Services Resource Framework including **WS-ResourceProperties, WS-ResourceLifetime, WS-RenewableReferences, WS-ServiceGroup,** and **WS-BaseFaults** (OASIS) http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf with Oasis TC set up April 2004 and V1.1 Framework March 2004 http://www-106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf
  - Uses rich metadata to define stateful interactions – its use of SOAP header creates interoperability problems
- **ASAP** Asynchronous Service Access Protocol (OASIS)
- http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=asap with V1.0 working draft G June 2004 http://www.oasis-open.org/committees/download.php/7151/wd-asap-spec-01g.pdf
- **WS-GAF** Web Service Grid Application Framework (Arjuna, Newcastle University) http://www.neresc.ac.uk/ws-gaf/
  - Uses WS-Context to provide "opaque" (don't say much) stateful interactions

Metadata Catastrophe I

- We keep finding places where metadata can be transmitted to and from services
- WS-Addressing and WS-RF specify metadata in SOAP header of messages
- WS-Context similarly specifies both SOAP header and WS-Context context services as location of (temporary) metadata
- We have registries like UDDI of service data
- WS-MetadataExchange covers metadata stored in services
  - Service metadata is very common and often not explicitly called out e.g. WebDAV as in Apache Slide stores file metadata in addition to versioning information
- In addition, we have major source of one or more (federated) catalogs
- I think this confused situation will need to be addressed by some new dynamic metadata model

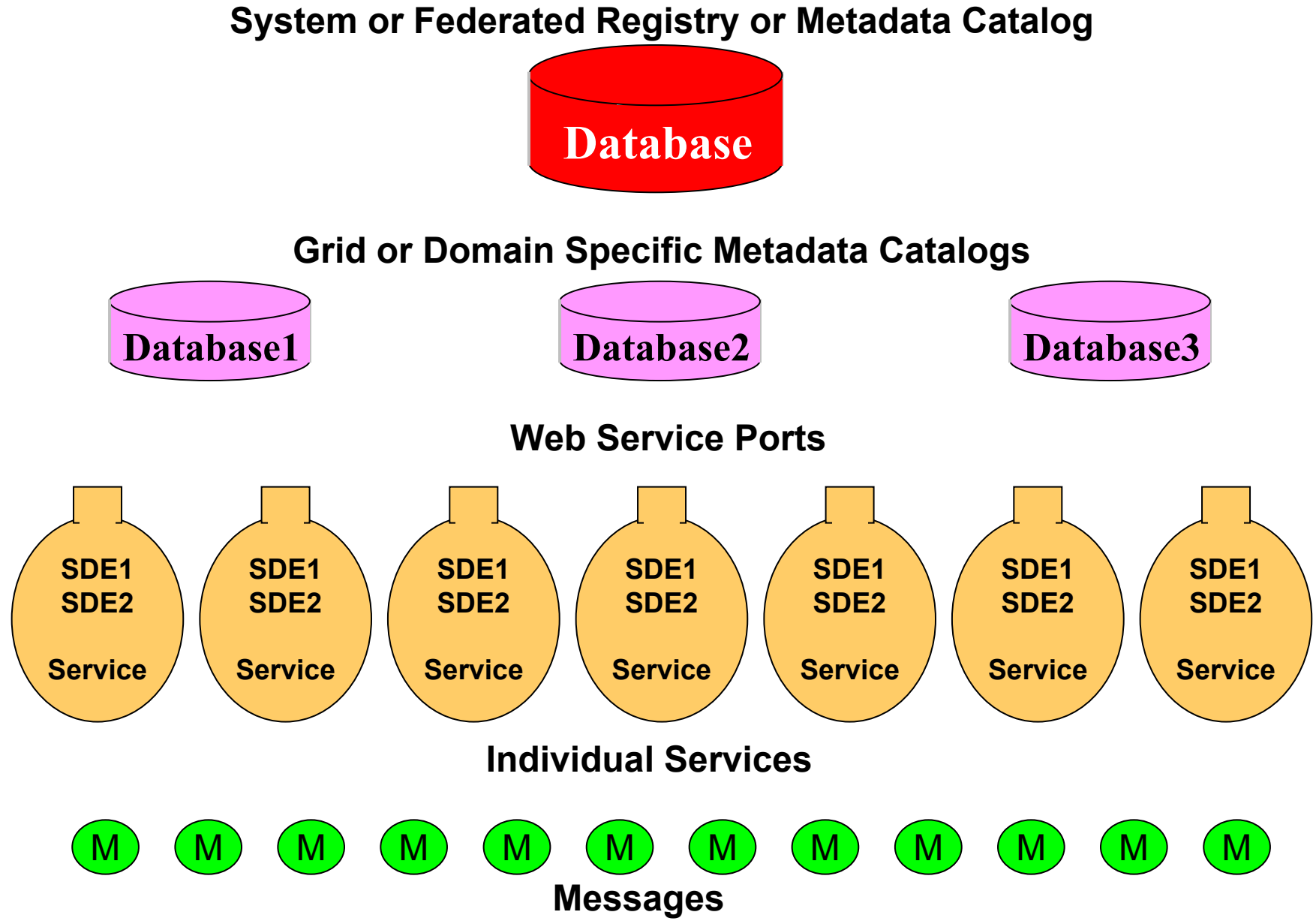
Metadata Catastrophe II

- There are large long term metadata catalogs associated with major applications/services
  - These are likely to remain as now based on traditional major database technology like Oracle MySQLK and DB2
- There are small but broadly available metadata catalogs
  - Globus MDS and EDG RGMA roughly address these
  - Semantic Grid enriched Service catalogs as in UDDI
- We need to implement UDDI in a distributed (federated) fashion and work around its non-intuitive schema but this seems straightforward
- All the problems occur for local and highly dynamic data where key issues are:
  - Consistency: If metadata stored in messages flowing around, how do we ensure consistency if it ever changes
  - Where is it: How do we decide where to look it up?
- My intuition is that best solution is highly dynamic lightweight database – doesn't really fit any proposal yet!

Metadata and Semantic Grid

- Can store in one catalog, multiple catalogs or in each service
  - Not clear how a coherent approach will develop
- Specialized metadata services like UDDI and MDS (Globus)
  - Nobody likes UDDI
  - MDS uses old fashioned LDAP
  - RGMA is MDS with a relational database backend
- Some basic XML database (Oracle, Xindice …)
- "By hand" as in current SERVOGrid Portal which is roughly same as using service stored SDE's (Service Data Elements) as in OGSI
- Semantic Web (Darpa) produced a lot of metadata tools aimed at annotating and searching/reasoning about metadata enhanced webpages
  - Semantic Grid uses for enriching Web Services
  - Implies interesting programming model with traditional analysis (compiler) augmented by meta-data annotation

Four Metadata Architectures

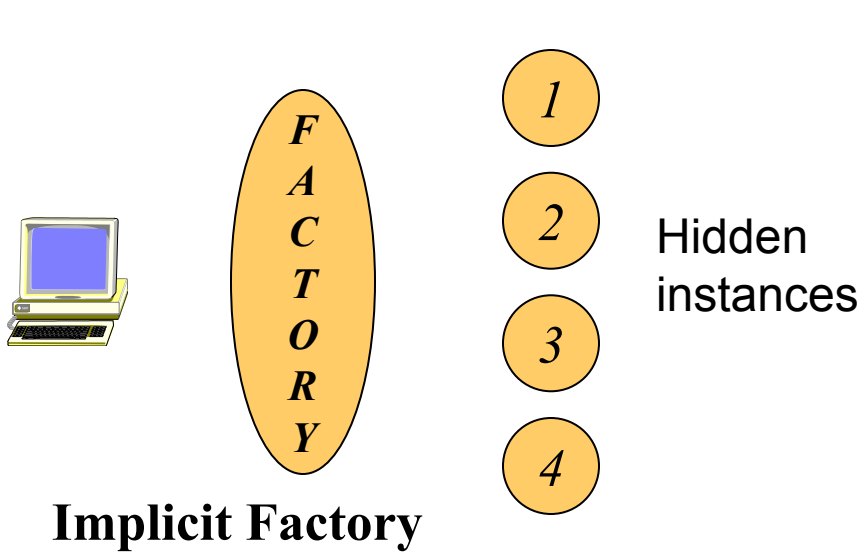


Stateful Interactions

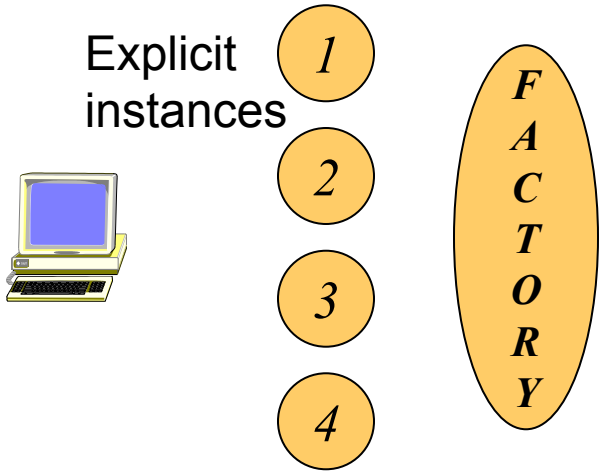
- **There are (at least) four approaches to specifying state**
  - **OGSI use factories to generate separate services for each session in standard distributed object fashion**
  - **Globus GT-4 and WSRF use metadata of a resource to identify state associated with particular session**
  - **WS-GAF uses WS-Context to provide abstract context defining state. Has strength and weakness that reveals less about nature of session**
  - **WS-I+ "Pure Web Service" leaves state specification the application – e.g. put a context in the SOAP body**
- **I think we should smile and write a great metadata service hiding all these different models for state and metadata**

Explicit and Implicit Factories

- Stateful interactions are typified by amazon.com where messages carry correlation information allowing multiple messages to be linked together
  - Amazon preserves state in this fashion which is in fact preserved in its database permanently
- Stateful services have state that can be queried outside a particular interaction
- Also note difference between implicit and explicit factories
  - Some claim that implicit factories scale as each service manages its own instances and so do not need to worry about registering instances and lifetime management



Implicit Factory



Explicit Factory

WS-I+ Grid Interoperability Profile

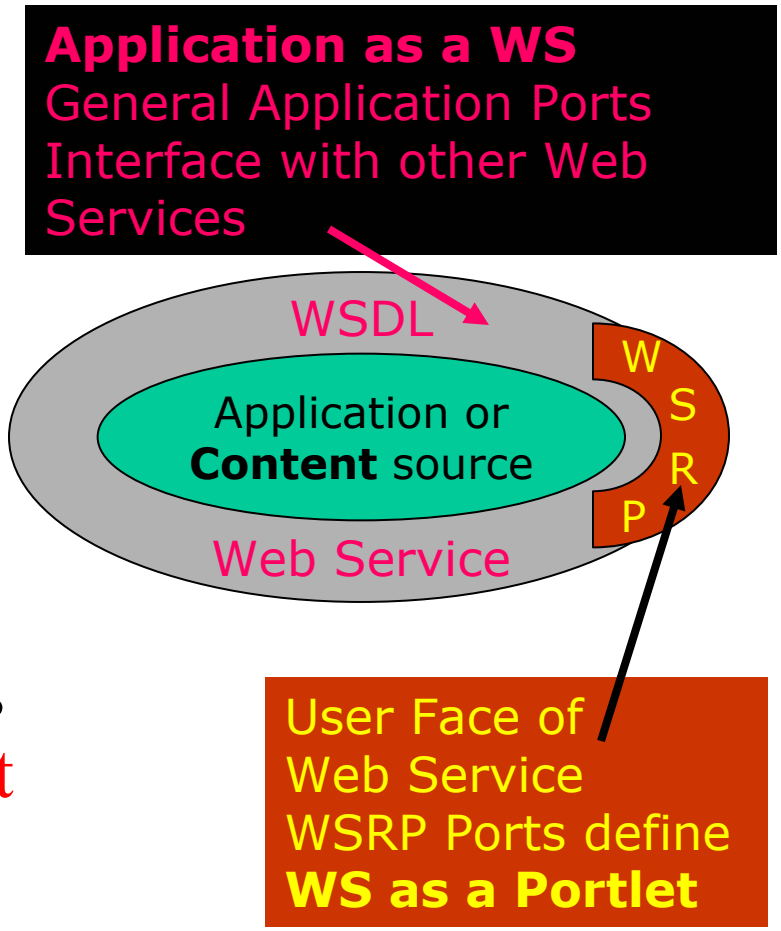
- WS-I identifies XSD, WSDL1.1, SOAP1.1, UDDI
- WS-I+ adds minimum additional capabilities to WS-I to allow development of Grid Services
  - BPEL for workflow
  - WS-Addressing for virtualization and richness of messaging
  - WS-ReliableMessaging/Reliability to provide basis for fault tolerance
- And it expects progress in
  - Security – need to understand better as Web Services are not settled down and many large projects like Shibboleth
  - Notification – hopefully IBM and Microsoft will agree
- while use of portlets will be encouraged (later)
- Open Middleware Infrastructure Institute http://www.omii.ac.uk/

Web Service User Interfaces

- **WSRP** Web Services for Remote Portlets (OASIS) OASIS Standard August 2003 http://www.oasis-open.org/committees/download.php/3339/wsrp-specification-1.0-cs-1.0-rev3.pdf
- **JSR168:** JSR-000168 Portlet Specification for Java binding (Java Community Process) October 2003 http://www.jcp.org/aboutJava/communityprocess/final/jsr168/
  - GridSphere, Jetspeed and uportal are or will be JSR168 compliant and this gives portlet architecture with aggregation portals

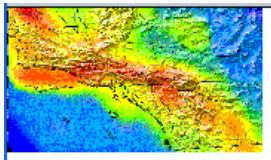
Web Services as a Portlet

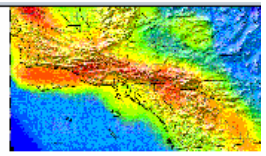
- Each Web Service naturally has a user interface specified as "just another port"
  - Customizable for universal access
- This gives each Web Service a Portlet view specified by WSRP (Web services for Remote Portals) or JSR168
- So component model for resources "automatically" gives a component model for user interfaces
  - When you build your application, you define portlet at same time



Web Services have other ports to interact with other Web Services

Collage of Portals





GEM Applications | Noahsark | Admin | Danube | Solar | Grids | SERVO Files | Fault DB

GEM Portal for Data Services

Code Selection

U.S. Department of Energy National Collaboratories

Fusion Grid Portal Demo

Resources View | Grid View | GPIR Machine Summary | Jobs | Manage Files | SRB | Logbook | Monitor | TRANSP

SRB

MySRB

MySRB Login Page
Use Ticket-based Access
Change MySRB Password
Forgot MySRB Password
Register As New User
MySRB Online Help

MySRB Login

SRB User Name: demouser
SRB User Domain Name: npaci
SRB User Password: ••••••
SRB Port Number: 5544
SRB Host: srb.sdsc.edu

Please exit SRB once you have finished

LogIn

Session will timeout in: 60 minutes

MySRB - A Quick Introduction

- What is MySRB? MySRB is a web-based browse and search interface to the Storage Resource Broker (SRB) developed at the San Diego Supercomputer Center (SDSC). The SRB facilitates information sharing by allowing users (1) to access files stored on heterogeneoues resources including disks, tapes and databases on different machines through logically organized catalogs; and (2) to manage and share data collections in a secure manner.
- Who can use MySRB? Currently, MySRB is restricted to users who have computer accounts at SDSC, and who have registered for MySRB. If you are not at SDSC and would like to use MySRB please contact srb@sdsc.edu.
- Can I use MySRB from anywhere? Yes, except during registration. Once you are registered user of MySRB, you can access it from anywhere via web browsers. When registering, the browser must be running inside the SDSC firewall (i.e., IP address of 132.249.x.x).
- What about security? MySRB uses secure-http (https) protocol using 128-bit RSA authentication. As an extra degree of protection, your browser receives a unique session key when you login; once its time limit (default 60 minutes) expires, you must login again in order to continue.
- Additional Information can be found at: MySRB Online Help Page.
- Please send enquiries or complaints to srb@sdsc.edu.

Logbook

Summary Information on DIII-D Physics Experiments

Click here for status of current experiment

Click here for most recent summary

Browse by calendar Year: Month: 01

Search by date of experiment (in ISO format e.g. 19990219 = February 19, 1999, 200003 = March 2000, 2001 = all of 2001)
Normal Quick Search

Search by shot number

Search by miniproposal number

Search by SESSION_LEADER name (1993-present)

Search by string in summary.
Normal Quick Search

Submit SQL query (the sample query finds the most recent shot)

```
SELECT shot, time_of_shot
  FROM summaries
 WHERE shot = ( SELECT MAX(shot) FROM summaries )
```

View Search Tips

Go to D3DRDB Home Page

Click to submit either a Chief Operator summary, Physics Operator summary or a Session Leader summary

CERQUICK Summaries ZIPFIT Summaries

Feel free to send any questions, suggestions, comments, or concerns about this web page and the D3DRDB database to Justin Burruss (burruss@fusion.gat.com).

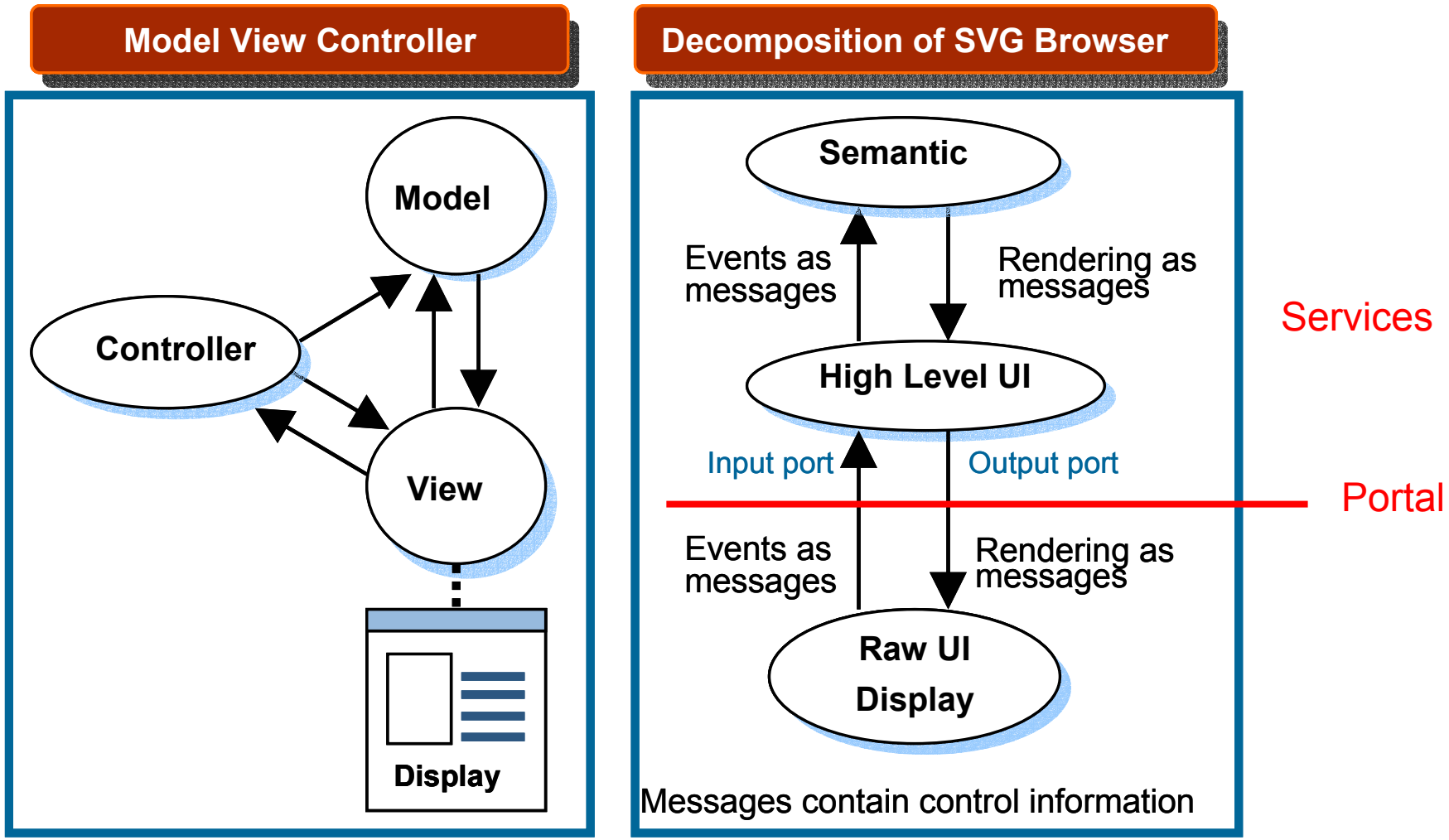
- CDLT
- OKC

Earthquakes – NASA
Fusion – DoE
Computing Info – DoD
Publications -- CGL

Issues in Portlets

- Current standards provide for a negotiation between clients and user facing Web service ports
- They do not address dynamic interfaces
  - That's why Java applets used as they internally support dynamic content
  - Used in audio video conferencing portlet
- They do not address communication between different portlets
- Rendering on clients is limited as constructs like HTML tables are not high technology
  - Better rendering engine desired

Portlets imply Message based MVC

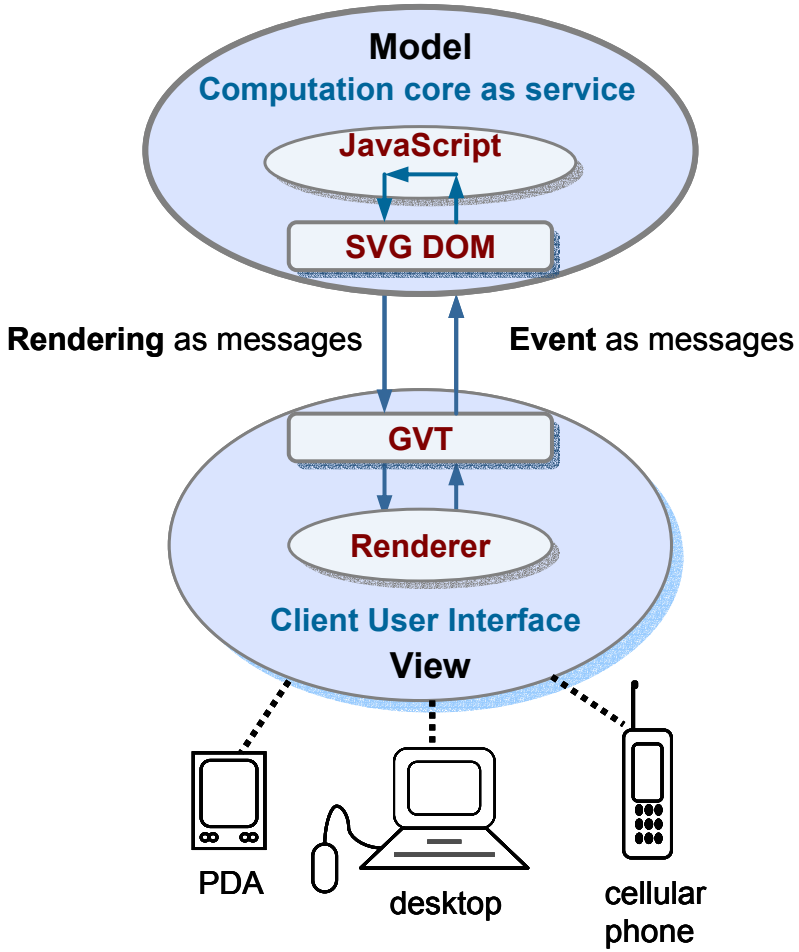


a. MVC Model

b. Three-stage pipeline

Can build desktop applications in this fashion

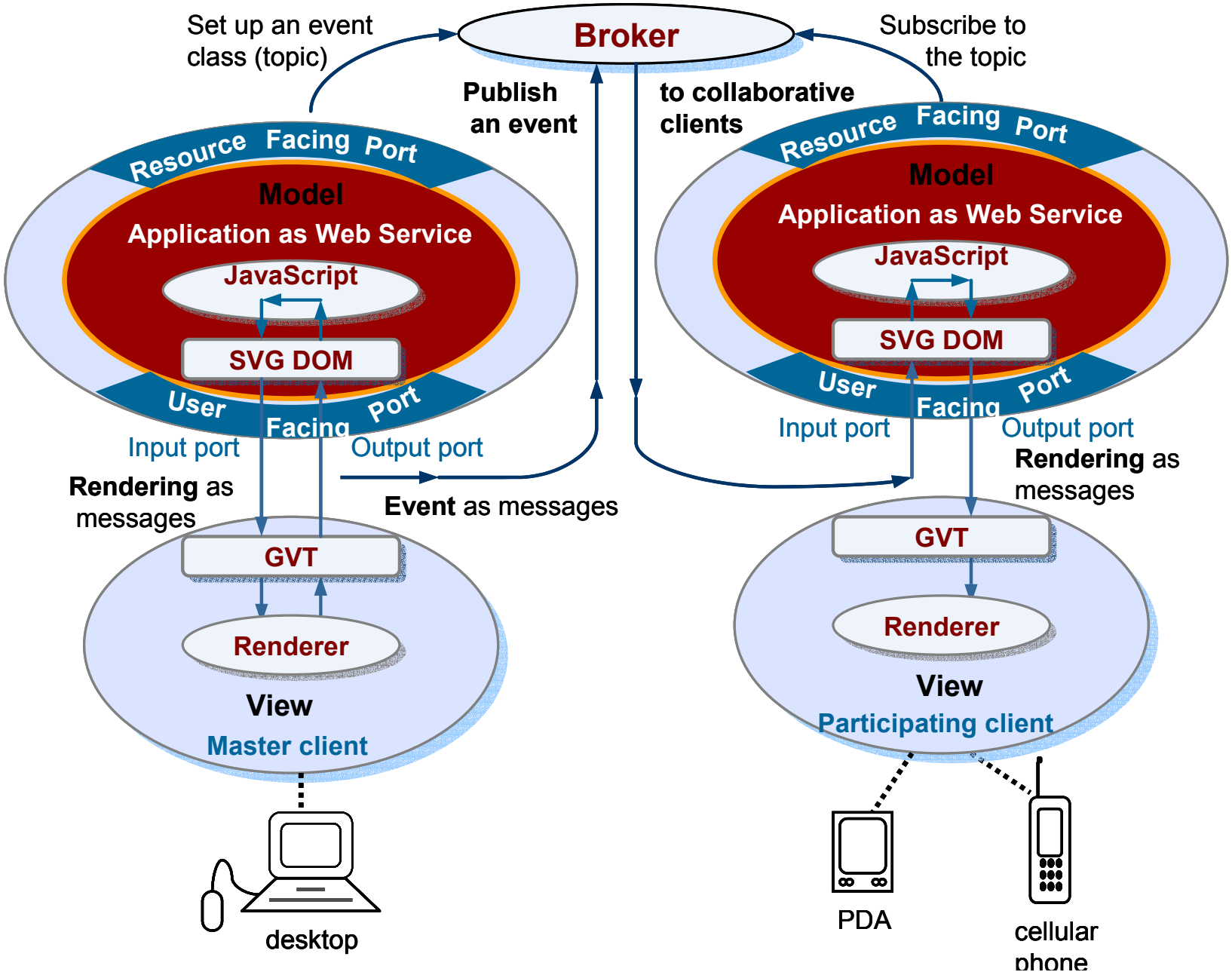
- Remember rule of millisecond – user interfaces don't notice a few (30) milliseconds
- Don't build complex clients
- Build Services!



Messages contain control information

SVG browser derived from message-based MVC

Shared Input Port Collaboration with Web Services



Some more general Grid Service Issues

Important Higher Level Services

- There will be uncountable services associated with particular applications but there are some services of broad applicability
- Accounting and higher level authentication and authorization security/privacy services
- Data movement such as GridFTP and GridRPC
- Metadata, Logging (small data items)
- Data Information and Knowledge Repositories – OGSA DAI with database (any type) or file access
  - Includes capabilities like WebDAV or just "Grid NFS"
- Computing services
  - Job Submittal, Status
  - Scheduling as in Condor, PBS, Sun Grid Engine
  - Links to MPI

Virtualization

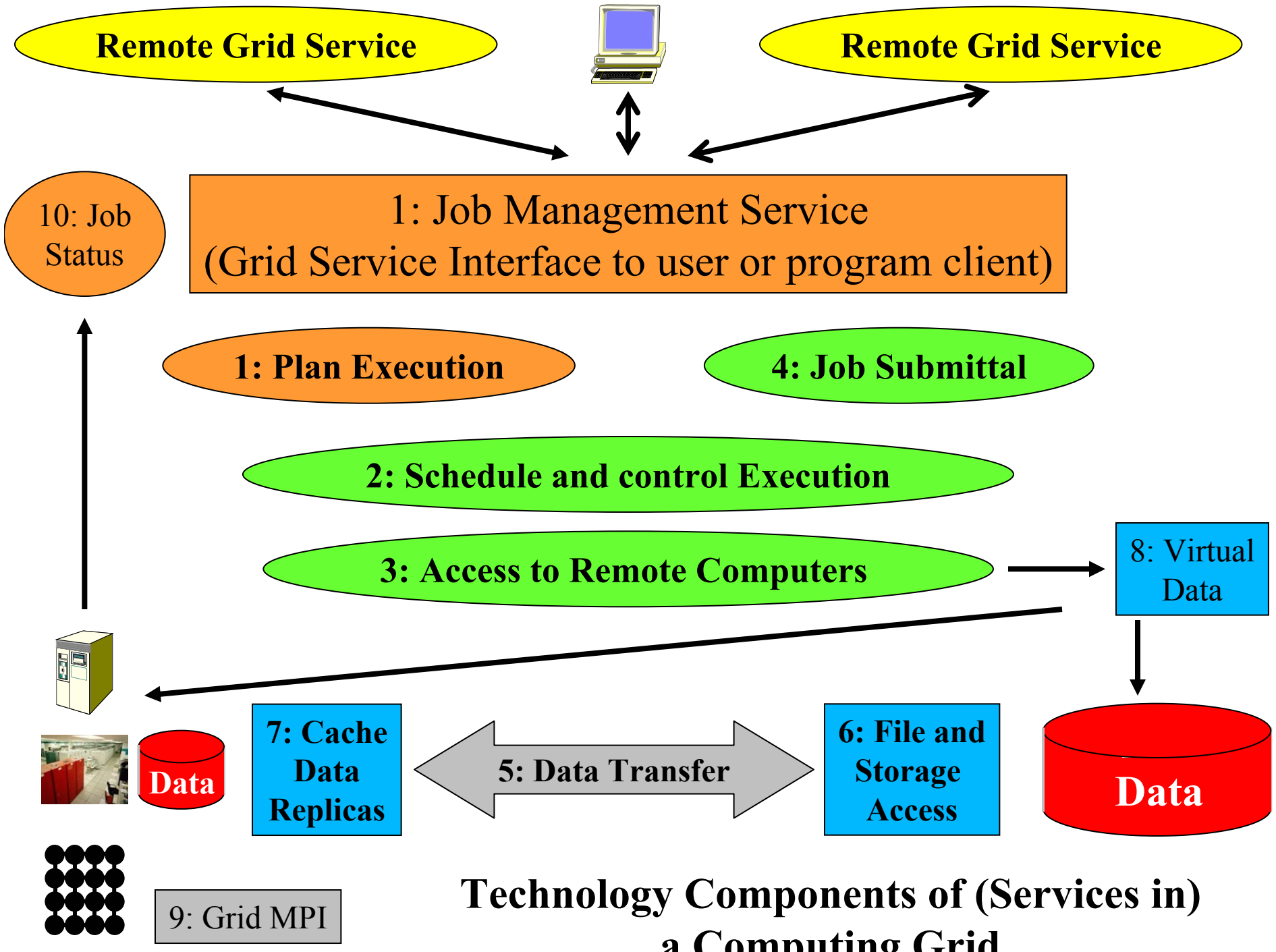
- The Grid could and sometimes does virtualize various concepts – should do more
- Location: URI (Universal Resource Identifier) virtualizes URL (WSAddressing goes further)
- Replica management (caching) virtualizes file location generalized by GriPhyn virtual data concept
- Protocol: message transport and WSDL bindings virtualize transport protocol as a QoS request
- P2P or Publish-subscribe messaging virtualizes matching of source and destination services
- Semantic Grid virtualizes Knowledge as a meta-data query
- Brokering virtualizes resource allocation
- Virtualization implies all references can be indirect and needs powerful mapping (look-up) services -- metadata

Special Challenges for Grids

- Representation of State
  - Stateless services and stateful interactions
  - Contextualization
- Factories – essential in object models but not directly present in service models
- Cross Administrative Access
  - Running a job is a dangerous service
  - Running a particular job (e.g. the Gaussian Service) is not very dangerous but currently this service model of simulation is not very common
- Include high performance computers in Grid
- Should use streams (which can be very high volume) and not write files
  - Need schedulers etc. with stream abstraction

Issues and Types of Grid Services

- 1) Types of Grid
  - R3
  - Lightweight
  - P2P
  - Federation and Interoperability
- 2) Core Infrastructure and Hosting Environment
  - Service Management
  - Component Model
  - Service wrapper/Invocation
  - Messaging
- 3) Security Services
  - Certificate Authority
  - Authentication
  - Authorization
  - Policy
- 4) Workflow Services and Programming Model
  - Enactment Engines (Runtime)
  - Languages and Programming
  - Compiler
  - Composition/Development
- 5) Notification Services
- 6) Metadata and Information Services
  - Basic including Registry
  - Semantically rich Services and meta-data
  - Information Aggregation (events)
  - Provenance
- 7) Information Grid Services
  - OGSA-DAI/DAIT
  - Integration with compute resources
  - P2P and database models
- 8) Compute/File Grid Services
  - Job Submission
  - Job Planning Scheduling Management
  - Access to Remote Files, Storage and Computers
  - Replica (cache) Management
  - Virtual Data
  - Parallel Computing
- 9) Other services including
  - Grid Shell
  - Accounting
  - Fabric Management
  - Visualization Data-mining and Computational Steering
  - Collaboration
- 10) Portals and Problem Solving Environments
- 11) Network Services
  - Performance
  - Reservation
  - Operations



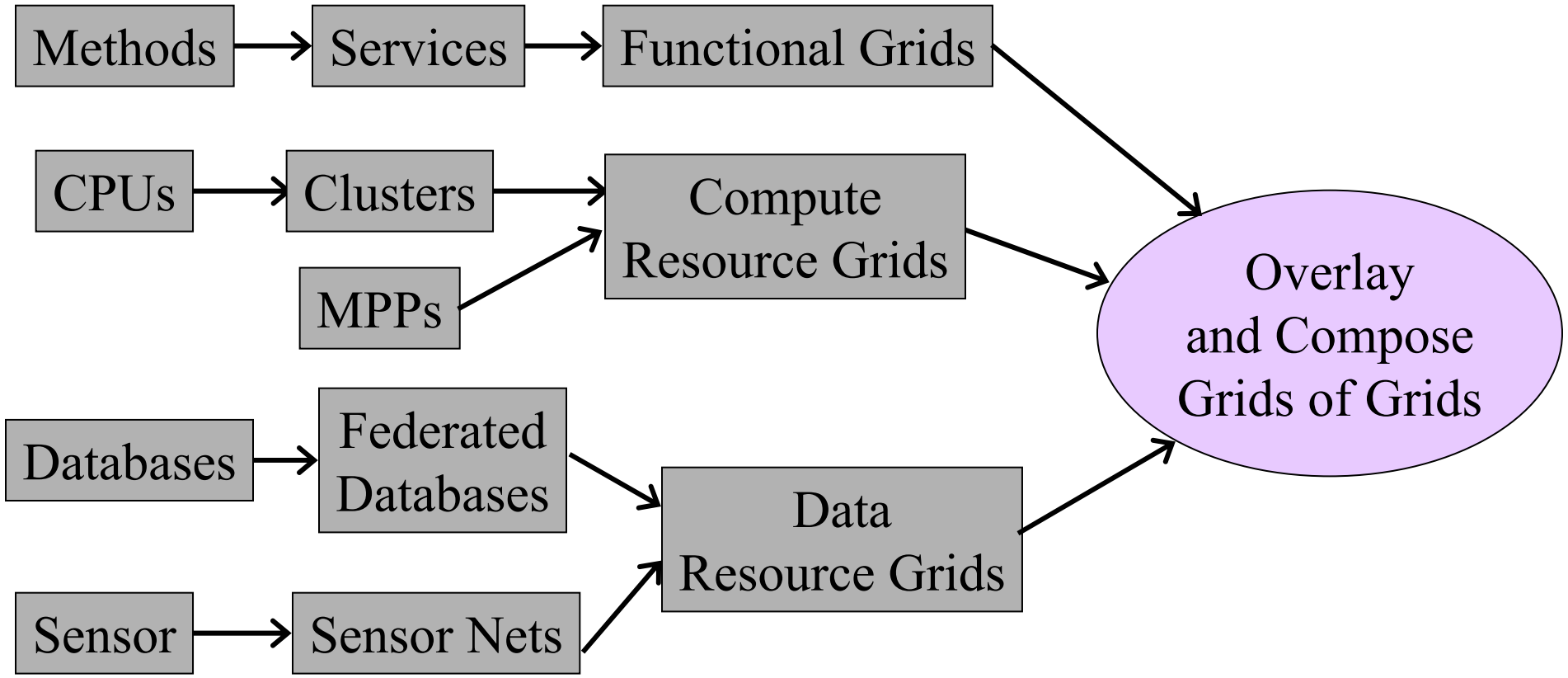
Technology Components of (Services in) a Computing Grid

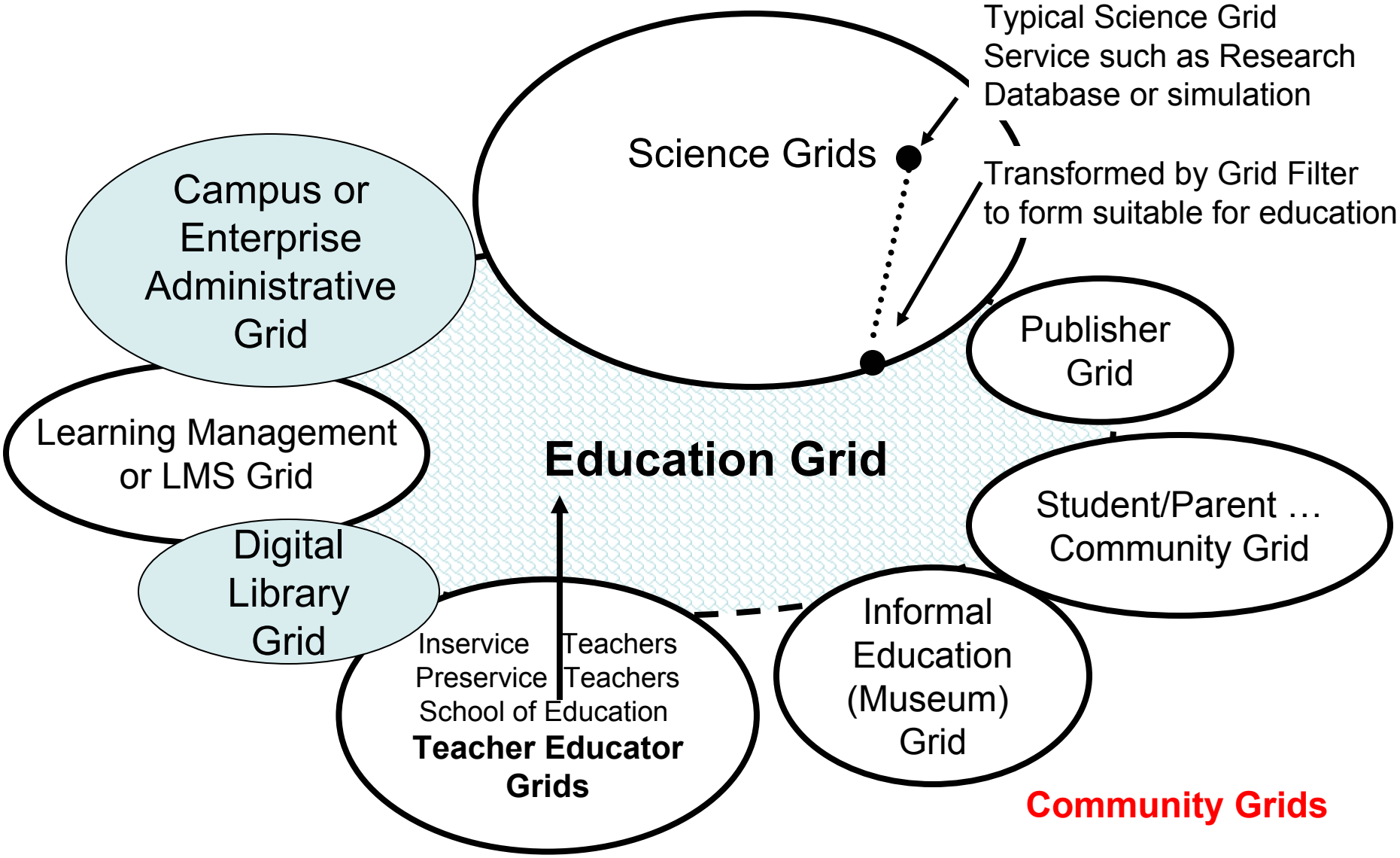
Taxonomy of Grid Operational Style

Name of Grid Style	Description of Grid Operational or Architectural Style
Semantic Grid	Integration of Grid and Semantic Web meta-data and ontology technologies
Peer-to-peer Grid	Grid built with peer-to-peer mechanisms
Lightweight Grid	Grid designed for rapid deployment and minimum life-cycle support costs
Collaboration Grid	Grid supporting collaborative tools like the Access Grid, whiteboard and shared applications.
RRR or Autonomic Grid	Fault tolerant and self-healing Grid Robust Reliable Resilient RRR

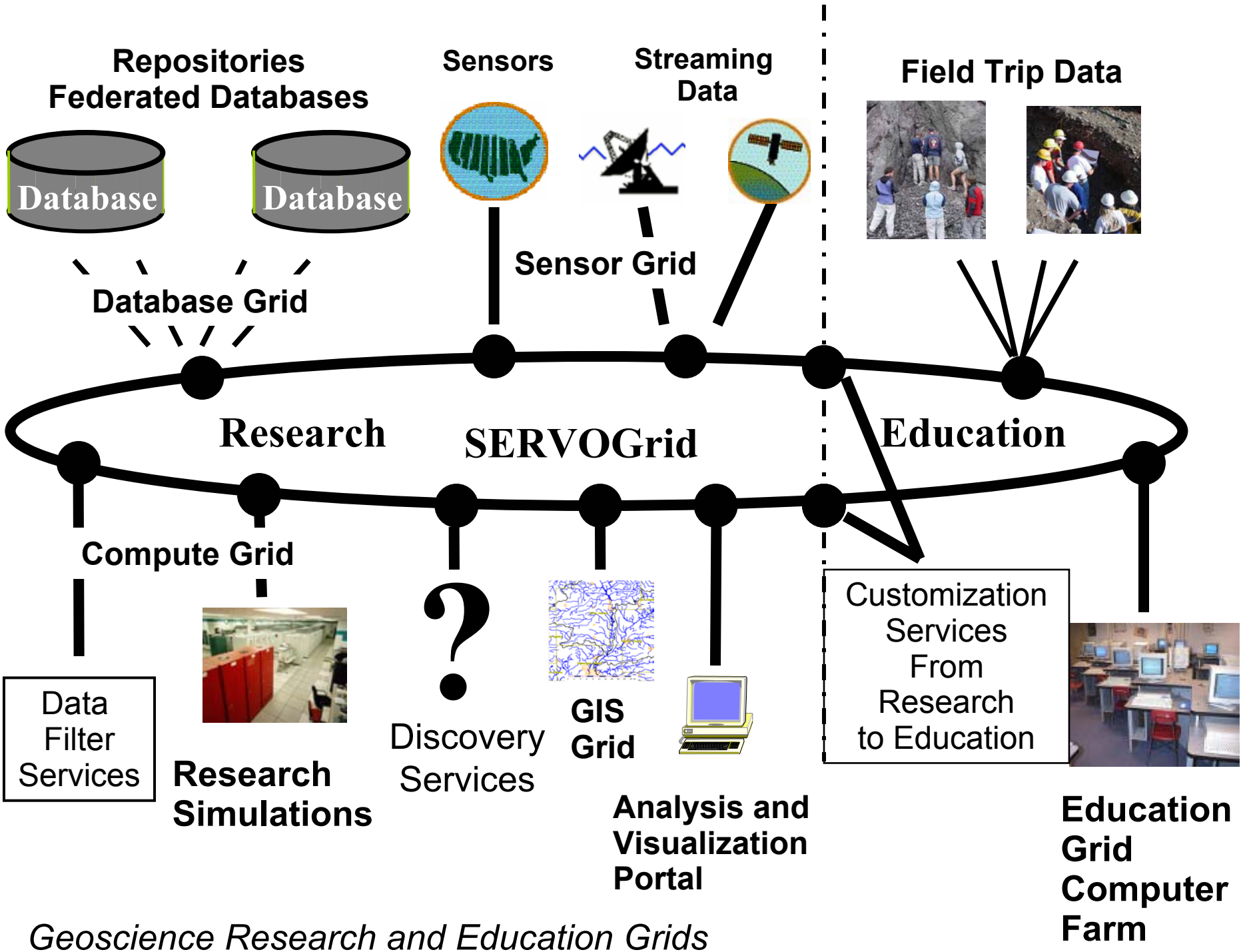
Grids of Grids of Simple Services

- Link via methods → messages → streams
- Services and Grids are linked by messages
- Internally to service, functionalities are linked by methods
- A simple service is the smallest Grid
- We are familiar with method-linked hierarchy
  Lines of Code → Methods → Objects → Programs → Packages

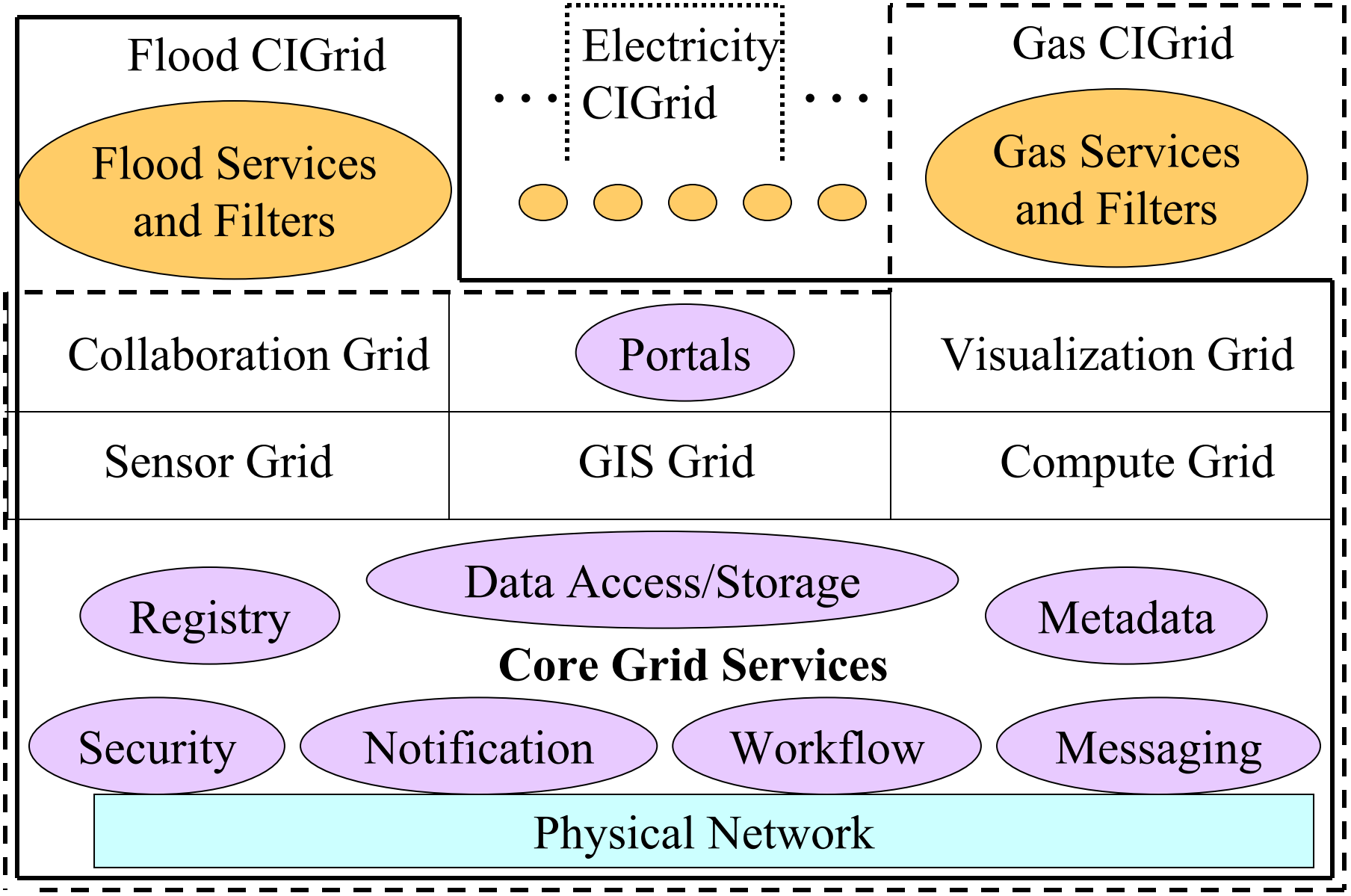




Education as a Grid of Grids



Geoscience Research and Education Grids



Critical Infrastructure (CI) Grids built as Grids of Grids