

WS DISPATCHER: Peer-to-peer Web Service and Firewalls

September 25 , 2004

Version 0.5

Alexandre di Costanzo (INRIA)
Aleksander Slominski (IU Extreme! Lab)

Goals

- **Support reliable long running conversation through firewalls of Web Service peers that have no accessible endpoints (behind firewalls, inside browsers)**
- Possible solution: asynchronous communication
 - Use of reliable messaging (WS-RM)
 - Use WS-Addressing (WSA) to identify and give logical names to service and client peers
 - Message level security can be added (wss4j)
 - Allows creation of session and in particular use of WS-SecureConversation
 - *Allows single sign-on (WS-Trust?!)*

Supporting: RPC? Messaging?

	RPC based service (synchronous request-response)	Messaging based service (asynchronous one-way)
RPC client (synchronous request-response)	Limited: RPC connection is forwarded and client may timeout, hard to keep long connection opened – resource hog	Very limited (will not work – message reply may come much later)
Messaging client (asynchronous one-way)	RPC service is bottleneck (translating Messaging to RPC ...)	Uncoupled (client and services can be separated in time)

Request-response (RPC case)

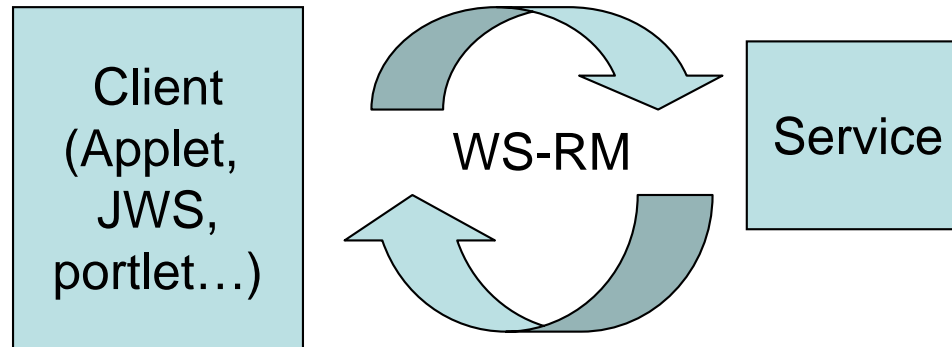
- Client opens connection to service location:
 - http://dispatcher_host/service/foo
- Dispatcher rewrites HTTP headers and (after checking security etc.) opens HTTP connection to actual service (possibly doing load balancing):
 - http://some_internal_host/axis/foo
- Dispatcher waits for response from service and then send it back to requestor on the same HTTP connection
 - Timeouts are big problem especially if multiple dispatchers are chained!

Asynchronous WSA messaging

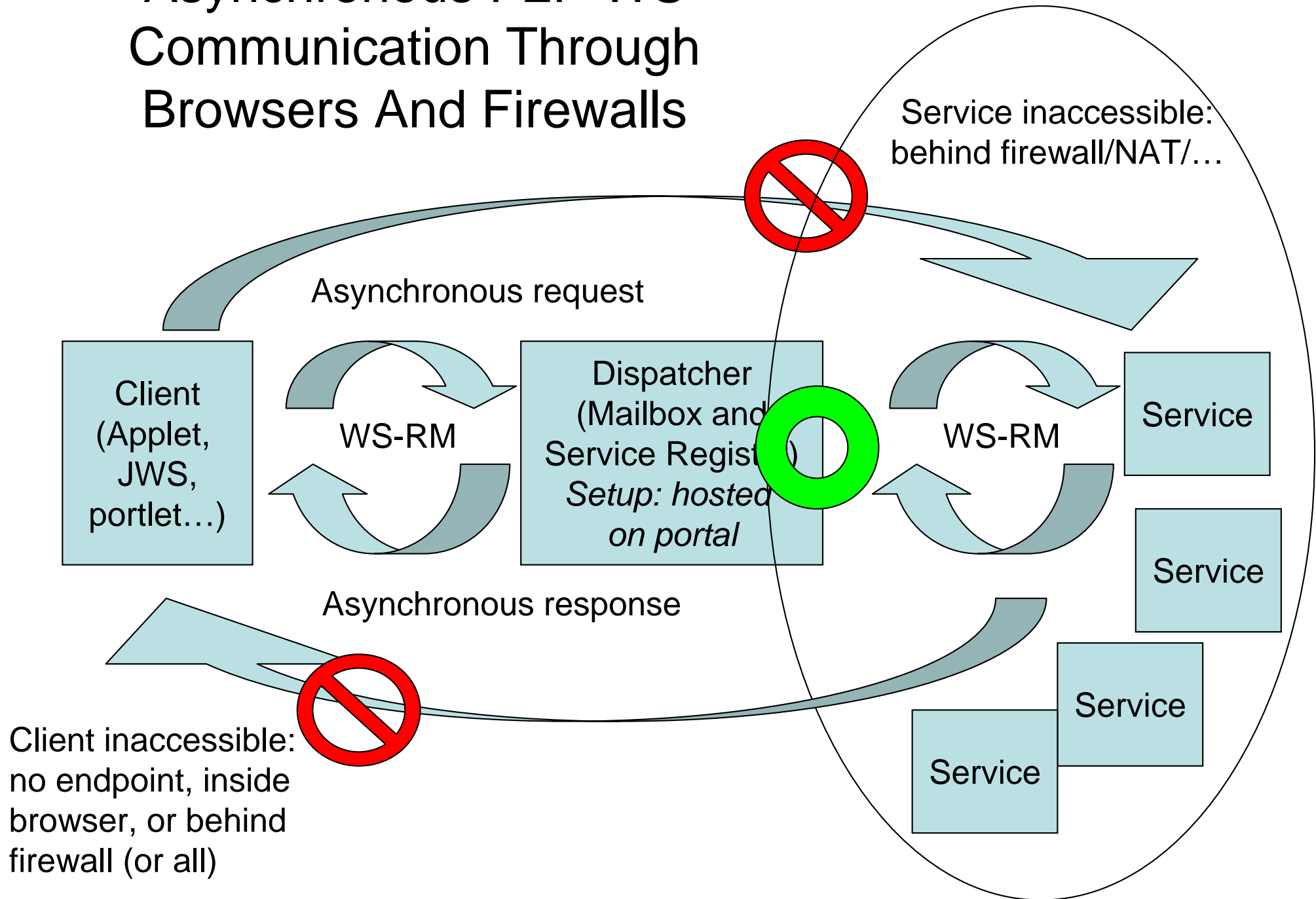
- Clients send message to dispatcher
 - One-way HTTP is used, HTTP response with empty body is returned immediately to client when message is accepted
- Dispatcher examines WSA:destination and if no service exists and policy is set to notify about failure sends fault to WSA:replyTo/faultTo
- Dispatcher rewrites WSA:destination/replyTo in message and sends it to actual service
 - This can be load balanced, other transport may be used (JMS internally), and multiple messages can be batched in queues
- **Optional:** only if there is actually response (or fault) generated by service
- Dispatcher will accept internal service response when it receives messages with special WSA:destinaton corresponding to rewritten WSA:replyTo
- Dispatcher is rewriting again WSA:destination and WSA:replyTo and sends message to client (peer)
 - In case of applets message is stored in P.O. box (see next slides)

Asynchronous P2P WS Communication Through Browsers And Firewalls

Logical View

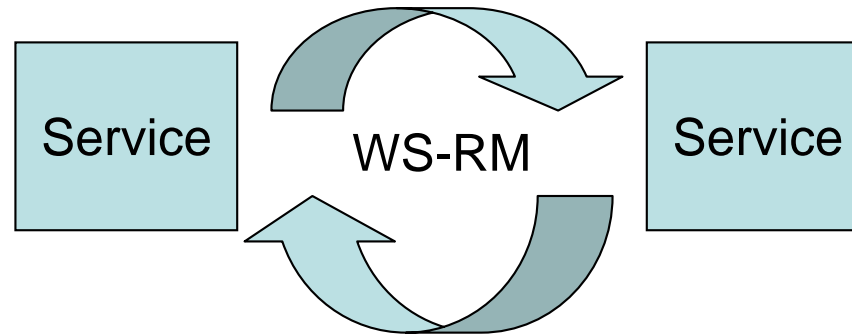


Asynchronous P2P WS Communication Through Browsers And Firewalls

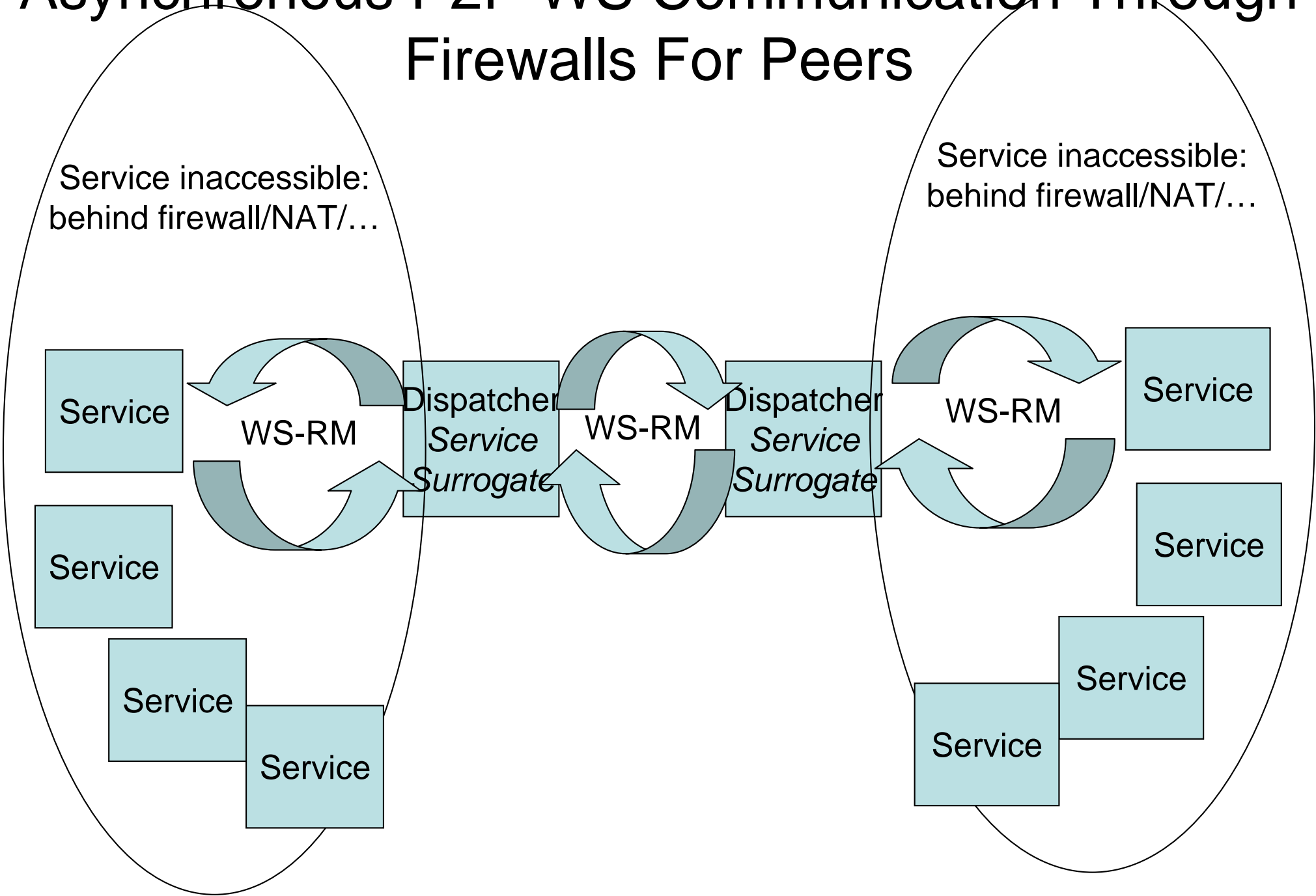


Asynchronous P2P WS Communication Through Firewalls For Peers

Logical View



Asynchronous P2P WS Communication Through Firewalls For Peers



Dispatcher Functions

- HTTP Proxy / Router
 - Tunnels SOAP-RPC/HTTP invocations
- Service Registry
 - List of publicly accessible service metadata (WSDLs)
- Mailbox
 - Allow sending/receiving message for peers behind firewalls or disconnected peers (when they reconnect)
 - **Forwarding service** and **PO boxes**

Forwarding service

- Trusted service can register where messages should be forwarded
 - Possible permanent logical address of service
 - Service is long running
 - Messages may be “hold” for some time
 - Load balancing may be supported

Registry of services

- List of *permanent* Web Services that are behind firewall and made accessible through dispatcher
- Dispatcher provides “logical” **and permanent** address for services
- Dispatcher translates logical address to known physical location
- Dispatcher can also “hold” messages when Web service is not accessible (or migrating)
- Dispatcher can load-balance service
- **Yellow book**

P.O Box service

- PO Box: Allows to create temp. peer endpoint for client
 - By default: creates temporary address that expires after some time
 - Allows to receive messages by authorized client (pick-up)
 - Allows to publish messages to address (may do security checks)
 - Client can migrate and re-connect later to get messages

Conclusion

- Dispatcher was implemented both for HTTP forwarding/proxy method and using asynchronous WS-Addressing
 - Worked very well making available services through portal (tested for high loads of hundreds of connections)
- Implementing PO Box for applets requires special messages (polling) to pick up messages addressed to applet
 - Would it be possible to use WS-Enumeration?