External Data Representation & Indirect Messaging

WEB SERVICE COMMUNICATION

FRANK WALSH

Agenda

- External Data Representations
  - XML
  - JSON
- Indirect messaging
  - Group Communication
  - Publish/Subscribe
  - Queueing

External Data Representation

- Information in processes/programs held in Data Structures
  - E.g Array of Strings , Object instances,
- For one program to transmit information to another across a network, the corresponding data structure must be “flattened”
  - converted to a sequence of bytes before transmission and then rebuilt
- Sort of analogous to getting an ice cube through a funnel.
  - Turn the ice cube to water
  - Pass it through the funnel
  - Reconstruct the ice cube(freeze the water again)
  - How do you reconstruct the ice cube with exactly the same dimensions?

External Data Representation

- To pass data across a channel between two computers:
  - values are converted to an agreed external format before transmission. Values converted to the local form on receipt
  - The values are transmitted in the sender's format, together with an indication of the format used, and the recipient converts the values if necessary
- An agreed standard for the representation of data structures and primitive values is called an ***external data representation***

External Data Representation

- Marshalling
  - the process of taking a collection of data items and assembling them into a form suitable for transmission in a message
- Unmarshalling
  - The reverse of above
- Next we'll look at 2 external data representation machanisms
  - XML
  - JSON

External Data Representation
XML

- eXtensible Markup Language(XML)
- Same heritage as HTML(but XML is NOT HTML)
- XML data items are tagged with ‘markup’ strings
  - used to describe the logical structure of the data
- XML has many uses(as you will see later). For now we confine ourselves to external data representations
- Has many cool features including
  - Extensible
  - Textual
  - Kind of human readable and machine readable…

XML

namespace

```
<person id="123456789">
                    <name>Smith</name>
                    <place>London</place>
                    <year>1984</year>
                    <!-- a comment -->
</person >
```

```
<person pers:id="123456789" xmlns:pers =
"http://www.cdk5.net/person">
          <pers:name> Smith </pers:name>
          <pers:place> London </pers:place >
          <pers:year> 1984 </pers:year>
</person>
```

- Above shows XML definitions of the Person structure.
  - As with xHTML, tags enclose character data.
  - Tags : <name>, <place>,<year> data:”Smith”, “London”…
- Namespaces provide a means for scoping names

External Data Representation JSON

- JavaScript Object Notation
- Lightweight text-based open standard designed for human readable data interchange.
- Can represent simple data structures and associative arrays.
- Good for serializing and transmitting structured data across a network

JSON

- JSON is often used in Ajax techniques
- Often seen as low overhead alternative to XML
- Application programming interfaces(APIs) exist for most programming languages

```
{
   person:{
      id:123456789,
      name:'Smith',
      place:'London',
      year:1984
   }
}
```

Object Serialisation vs. XML vs. JSON

- XML can include type information(using XML schema)
  - XML designed to be “platform independent”, open standard
  - most programming languages, including Java, provide processors for translating between XML and language-level objects
- JSON
  - More straight forward than XML
  - In XML, same data can be represented several ways(example in class)
    <person id=“123456779” name=“smith” place=“london” year=“1984” />
    Same representation in JSON
  - JSON has one straight forward way

Indirect Messaging

Using the "Middleman"

- Communication between processes using an intermediary
  - Sender → "The middle-man" → Receiver
  - No direct coupling
- Up to now, only considered Direct Coupling
  - Introduces a degree of rigidity
- Consider…
  - What happens if client or server fails during communication in Direct Coupling?
- Two important properties of intermediary in communication
  - *Space uncoupling*
  - *Time uncoupling*

Space and Time uncoupling

	Time-coupled	*Time-uncoupled*
Space coupling	*Properties*: Communication directed towards a given receiver or receivers; receiver(s) must exist at that moment in time *Examples*: Message passing, remote invocation (see Chapters 4 and 5)	*Properties*: Communication directed towards a given receiver or receivers; sender(s) and receiver(s) can have independent lifetimes *Examples*: See Exercise 15.3
Space uncoupling	*Properties*: Sender does not need to know the identity of the receiver(s); receiver(s) must exist at that moment in time *Examples*: IP multicast (see Chapter 4)	*Properties*: Sender does not need to know the identity of the receiver(s); sender(s) and receiver(s) can have independent lifetimes *Examples*: Most indirect communication paradigms covered in this chapter

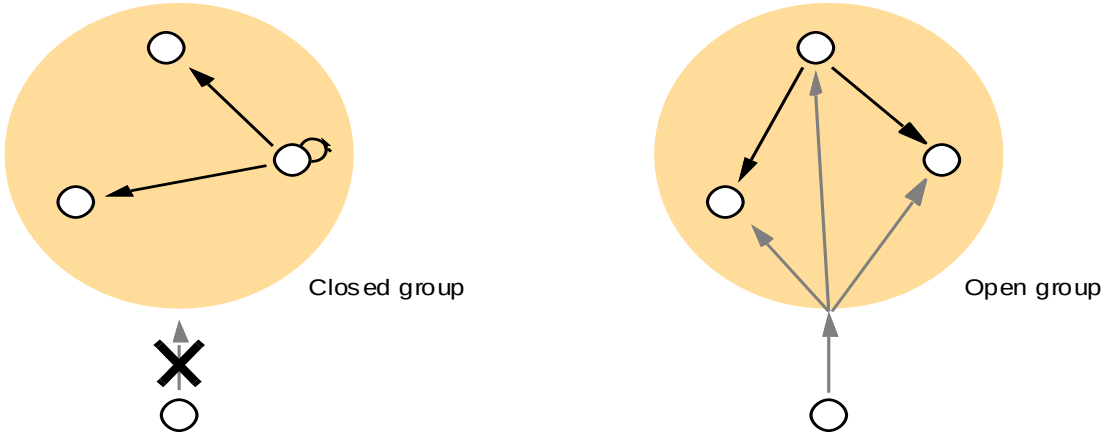
Time uncoupling vs. Asynchronous Comms

- Asynchronous communication
  - sender sends a message and then continues
  - No need to meet in time with receiver
  - Message buffered at receiver
- Time uncoupling
  - sender and receiver(s) can have independent existences
  - Receiver may not exist at the time communication is initiated

Group Communication

- Message is sent to a group
- Message is delivered to all members of the group
- Sender NOT aware of receiver identities
- Abstraction over multicast communication
  - Adds group membership, reliability, ordering
- Advantages:
  - reliable dissemination of information to potentially large numbers of clients
  - support for collaborative applications(online gaming)
  - range of fault-tolerance strategies
  - support for system monitoring and management,
- Programming model:
  - *aGroup.send(aMessage)*

Group Communication

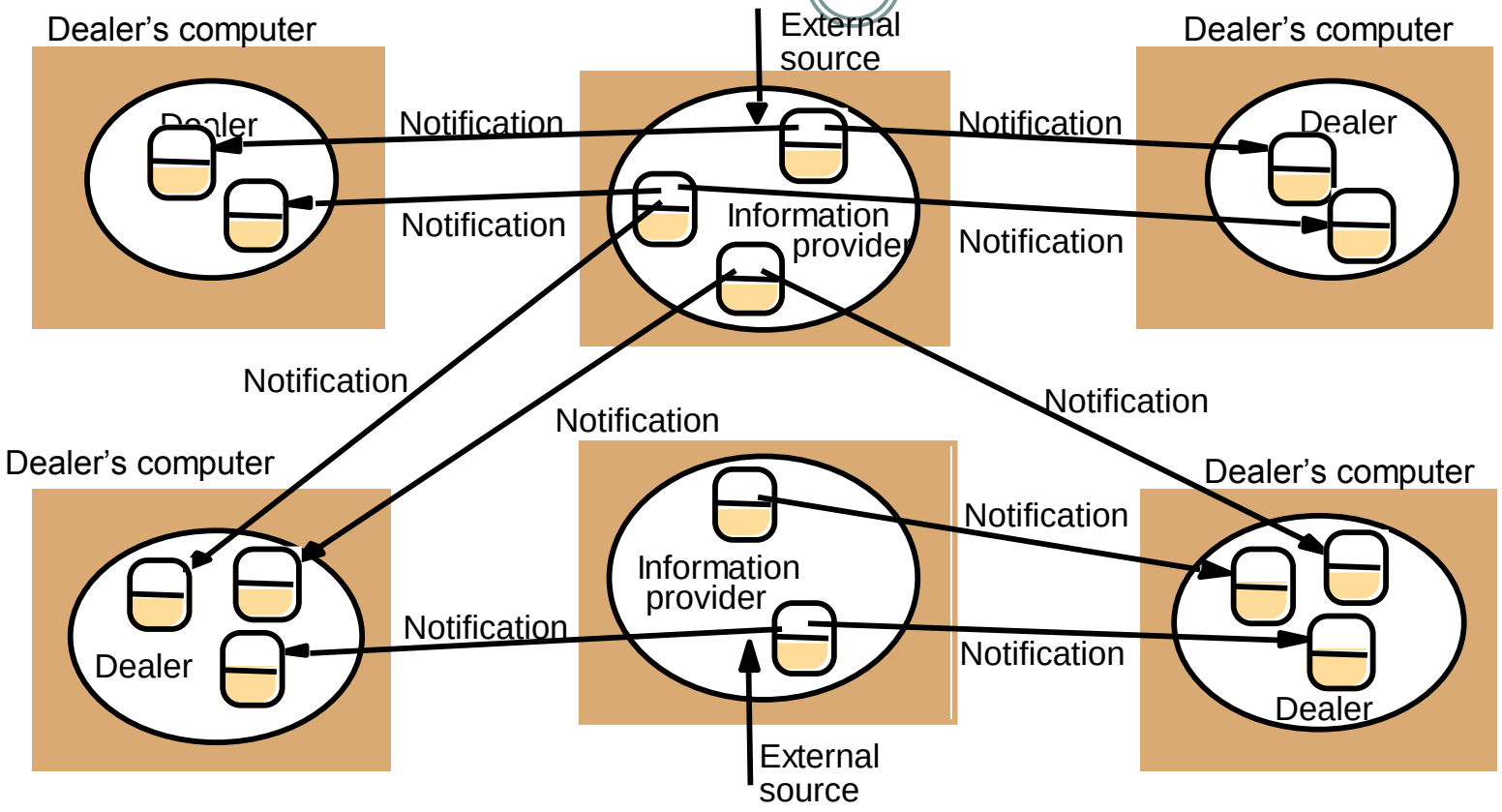


- Group is closed if only members of the group may multicast to it.
  - Example: coorperating servers
- Open group allows outside processes communicate
  - Example: delivering external events to interested groups(sensor data)

Publish-Subscribe

- Most widely used of all the indirect communication techniques
- Usually event based
  - Event published somewhere – pickup up by all subscribers
- Examples:
  - financial information systems
  - live feeds
  - ubiquitous computing(e.g. location events)
  - monitoring applications

Publish-Subscribe



- Publish-Subscribe: Dealing room system

Publish – Subscribe

- Publish-subscribe characteristics:
  - *Heterogeneity: distributed system that were not designed to interoperate can be made to work together*
    - *Example: Android based mobile device publishes location info. Smart home agent subscriber picks up events and acts accordingly(e.g. turn on heating when user gets home)*
  - *Asynchronicity:* Notifications are sent asynchronously to all subscribers – subscribers decoupled from publisher
    - *Example: subscriber can be a queue for a particular process. Queue is accessed by process as and when it can(could be busy at time of notification).*

Publish-Subscribe approaches

- Channel based:
  - publishers publish events to named channels. Subscribers subscribe and receive all events.
- Topic based
  - Each event associated with a “topic” or subject. Subscribers subscribe to a topic and recieve only topic events
- Content based
  - Similar to Topic based. Subscription based on range of event attributes. For example, subscriber might specify author attribute is “Fintan OToole” and category is “Finance”

Publish-Subscribe Example
Amazon Simple Notification Service(SNS)

- Scalable and flexible publish-subscribe cloud based service
- Topic-based approach
  - A topic is an “access point” – identifying a specific subject or event type – for publishing messages and allowing clients to subscribe for notifications
- Topic policies
  - Can limit who can publish messages or subscribe
  - specifying notification protocols(i.e. HTTP/HTTPS, email, SMS, SQS)
- Fairly Simple API for developers
  - **CreateTopic, Subscribe, Publish**
  - SDKs for all mainstream languages(Java, PHP, c# etc.)
  - More in labs….

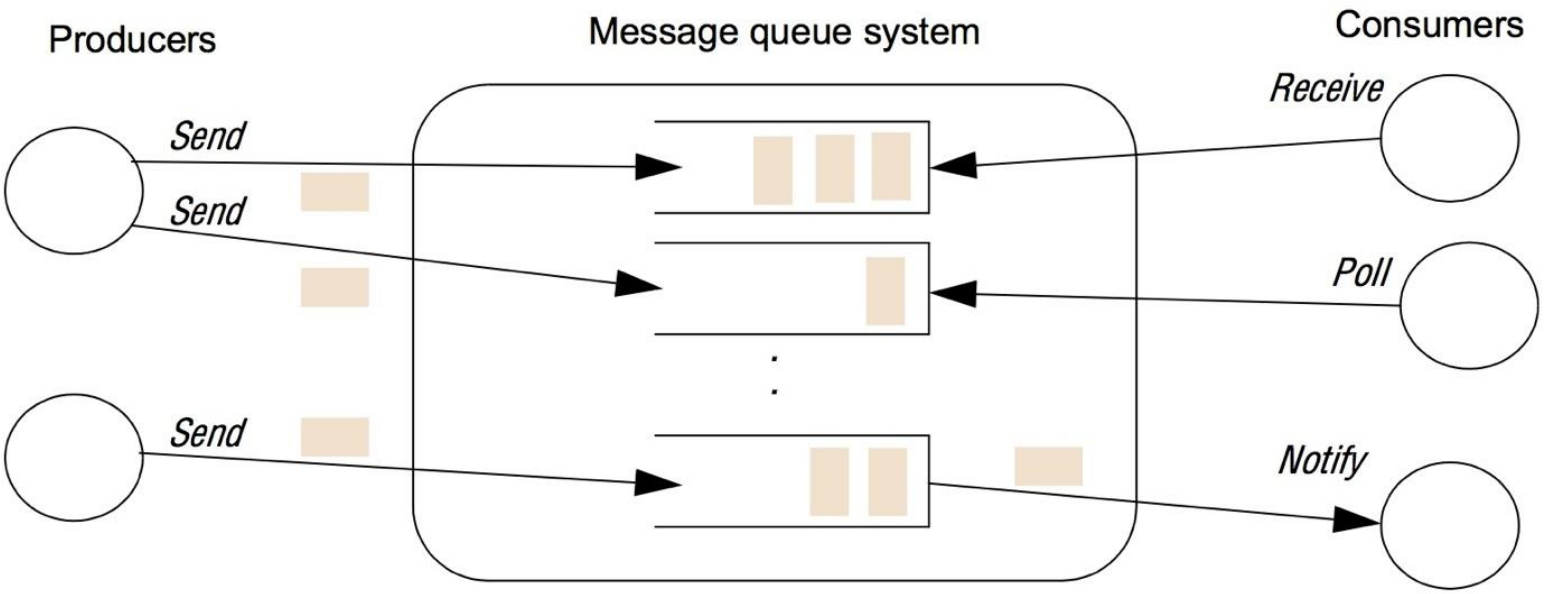
Publish-Subscribe Example: PubNub

- Scalable and flexible publish-subscribe cloud based service
- Topic-based approach
  - A topic is an "access point" – identifying a specific subject or event type – for publishing messages and allowing clients to subscribe for notifications
- Also provides for:
  - Push Notification
  - Storage and Playback (can behave like a Q)
  - Online Presence…
- Over 70 SDKs for all mainstream languages/frameworks (Java, JS, PHP…)
- You used the Node.js one in the lab.

Message Queues

- Publish-Subscribe is one to many
- Distributed Message Queues is point to point
- Distributes Message Queues often referred to as Message orientated Middleware(MOM)
- Examples
  - MQ Series
  - MS **MQMS**
  - Java Messaging Service

Message Queue



- Queues operate First in First out (FIFO)
- Modes of operation: Receive, Poll, Notify

Message Queue Applications

- Messages are persistent
  - Stored until consumed(although possible to set “time to live”)
- Supports reliable communication:
  - any message sent eventually received (validity)
  - message received is identical to the one sent
  - no messages are delivered twice (integrity)
- Can be used in conjunction with other middleware to implement transactions
  - Ensure all the steps in a transaction are completed, or the transaction has no effect at all (‘all or nothing’)
- Message Transformation
  - To support heterogeneity, transform messages between formats

Message Queues vs. Buffers

- Queues similar to buffers mentioned earlier in asynchronous message passing communications
- Buffers are implicitly associated with processes.
  - If the process goes down, the buffer will probably go down – no communication…
- Message queues are separate, third party, entities in the distributed system.
  - Receiving process can go down but queue will stay alive, keep queuing messages
- Queues facilitate for uncoupled, indirect comms.

References

- Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5 © Pearson Education 2012
- Lesson: All About Sockets : http://docs.oracle.com/javase/tutorial/networking/sockets/
- Amazon Web Services, SQS: http://aws.amazon.com/sqs/