Data Dissemination

Contents:

- Some Apps
- Nature of information
- Modes of interaction
- Data delivery options
- MOM
  - Addressing options

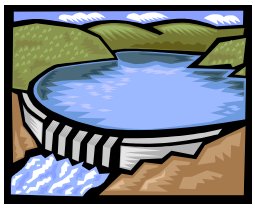
Mariano Cilia / cilia@informatik.tu-darmstadt.de

Motivated by Examples

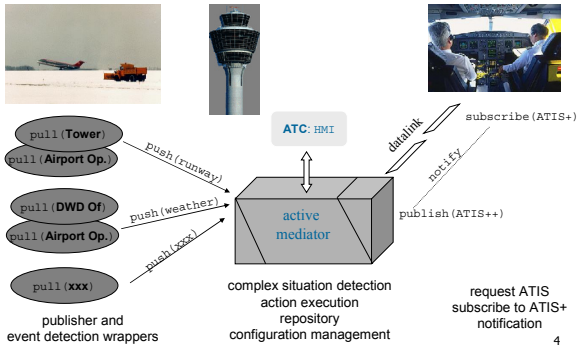
- Traffic management
  - Air-traffic control
  - Emergency vehicles
- Environmental
  - Concentration of chemicals
  - Dangerous emissions
  - Forest fires
  - Flood control

Example – Flood Control

- A combination of happenings (coming from sensor data) determines a situation
- Polling sensors
  - Too frequent
    - Consumption of resources
  - Too infrequent
    - Malfunction



Application scenario: ATIS



Examples (cont.)

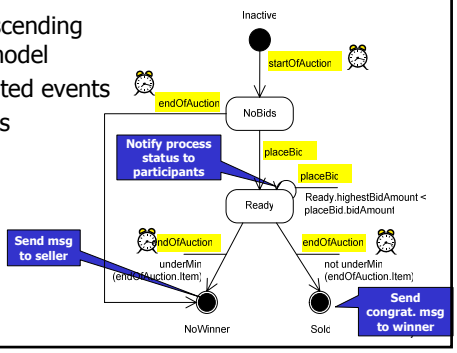
- Plant and reactor control
  - Equipment control
- Defense
  - Missile detection
  - Battlefield monitoring
- Workflow management

Examples (cont.)

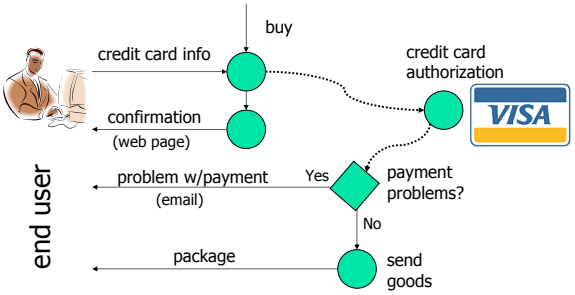
- Commerce
  - Inventory control
  - Supply Chain Management
  - Marketplaces
  - e-Auctions
  - Online shops

Example – e-Auctions

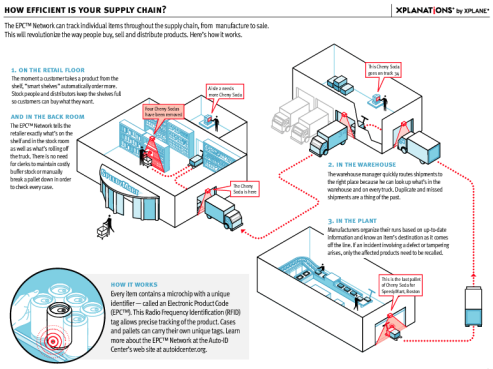
- Simple ascending auction model
- Time-related events
- Conditions



Example – Online shops



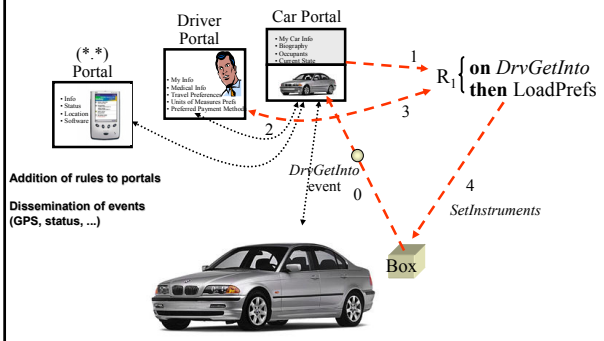
RFID – Supply Chain Mgmt (cont)



Examples (cont.)

- Personalization
  - User Interfaces
  - Services

Example - Personalization



Examples (cont.)

- Personalization
  - User Interfaces
  - Services
- Financial applications
  - Commodity trading
  - Currency trading
  - Stock trading

Example – Stock trading

- Sample
  - **ON** stock.Name=IBM
    IF stock.Price<20
    THEN call myBuy()
- High volume

13

Convergence of Technologies

- Ambient Intelligence and smart devices require continuous monitoring of events
- Miniaturization of sensors, ubiquitous deployment
- Context information for proper interpretation
- (Almost) complete reachability of individuals causes unbounded appetite for information
- Need to filter and interpret large amounts of heterogeneous and short-lived data
- Large distributed systems must detect and correct failures/exceptions (autonomic computing, ESCM, zero latency enterprise)

14

The Nature of Information

- Information *flows* from producer to consumer
  - info-pipes, broadcast disks, event streams, pub/sub
- Static view of information is a simplification
  - data flows into/out of high latency pool (database)

Mechanisms for access to static information (queries) different from those for accessing flow of information (subscription/filters)

15

Working Hypothesis

- Desintermediation/Reintermediation
- B2C, B2B, B2B2C, C2C, Portals, Markets (DotComs vs. NewCos), m-commerce …
- First generation e-commerce systems mapped existing applications 1:1 to new medium
- Next generation(s) will be based on flexible integration of services and components
- Flow of tasks and information
- How should (middleware) platforms look?

16

Modes of Interaction

1st generation

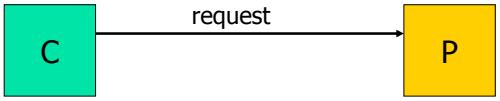
		Initiator of Interaction	
		Consumer	Producer
Knowledge about Counterpart	Full	Request/Reply	One-to-One Message
	No	Anonymous Request/Reply	Event-based dissemination

17

Modes of Interaction – R/R

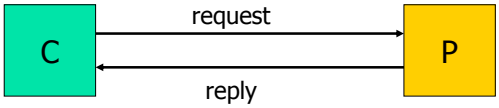
18

Modes of Interaction – R/R



- Known server

Modes of Interaction – R/R



Request/Reply

- Direct and synchronous communications
  - Enforces tightly coupling of comm. Parties
  - Impairs scalability
- Clients pull remote data sources
  - Trade off
    - Usage of data vs. data accuracy
      - Short polling interval → waste resource
      - Long polling interval → increase update latency
- Need for asynchronous and decoupled operations

Request/Reply (cont.)

- Simple
  - + imperative nature of C/S paradigm
  - + programming language abstraction
- Drawbacks
  - Point-to-point communication limits scalability
  - Polling limits accuracy of data
    - Unnecessary bandwidth consumption

Modes of Interaction

		Initiator of Interaction	
		Consumer	Producer
Knowledge about Counterpart	Full	Request/Reply	One-to-One Message
	No	Anonymous Request/Reply	Event-based dissemination

Anonymous R/R

- Consumer does not specify the provider
- Request is delivered to an arbitrary set of providers
- Identity of provider is unknown
- Load balancing

Modes of Interaction

		Initiator of Interaction	
		Consumer	Producer
Knowledge about Counterpart	Full	Request/Reply	One-to-One Message
	No	Anonymous Request/Reply	Event-based dissemination

1-to-1 message / Callback

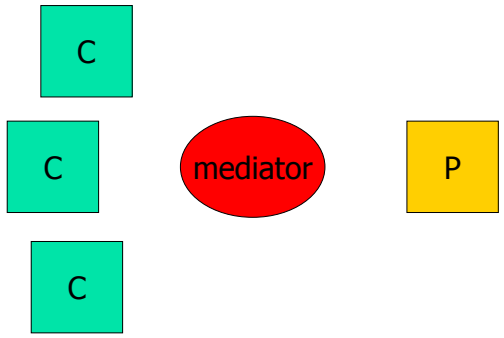
- Consumer registers interest with a known provider
- Provider repeatedly evaluates interests
  - when true → callback registered consumer
  - Responsible for managing list of interests and registered consumers
- One to one message
- Observer pattern

Modes of Interaction

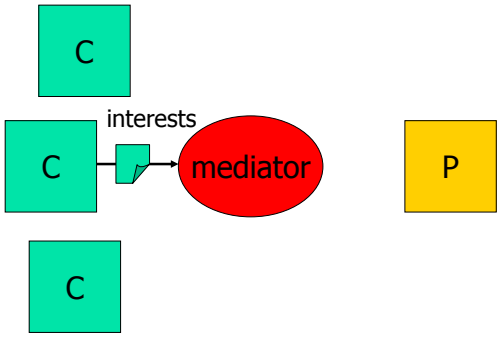
		Initiator of Interaction	
		Consumer	Producer
Knowledge about Counterpart	Full	Request/Reply	One-to-One Message
	No	Anonymous Request/Reply	Event-based dissemination

Needs a mediator

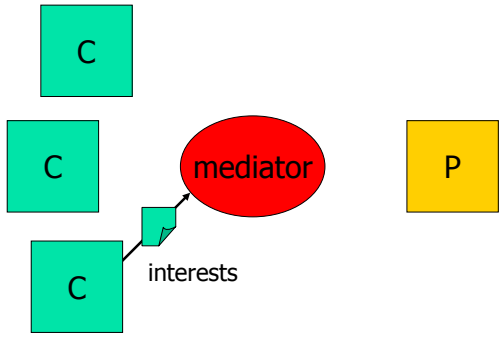
Modes of Interaction – Events



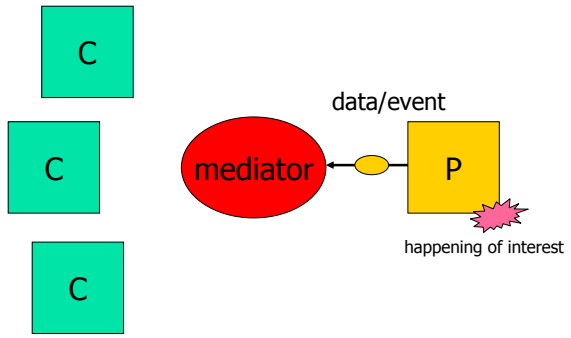
Modes of Interaction – Events



Modes of Interaction – Events

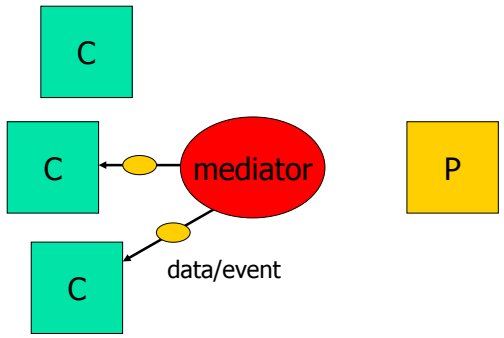


Modes of Interaction – Events



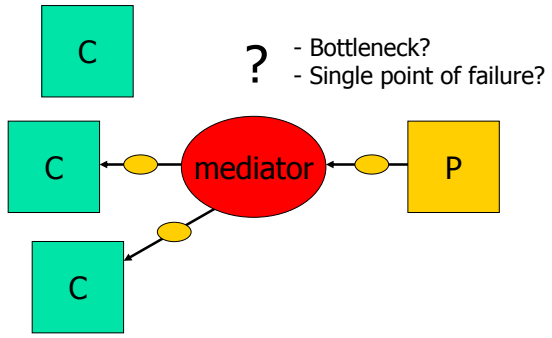
31

Modes of Interaction – Events



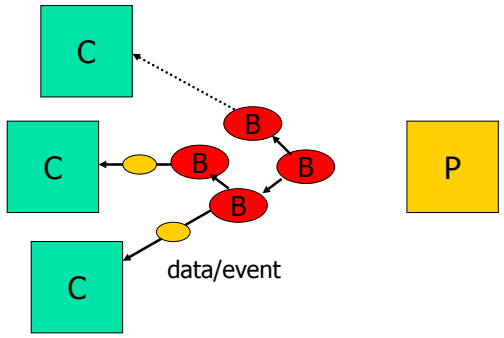
32

Modes of Interaction – Events



33

Modes of Interaction – Events



34

Events and Notifications

- Event
  - happening of interest at observed object(s)
- Notification
  - i) communication of event occurrence to interested recipients
  - ii) reification of observed event
- Notification Service (NS)
  - provides infrastructure to register for and deliver notifications
  - i.e. publish(), subscribe(), notify()

35

Event Notification - Patterns

- Observer: observable events
  - Event producers have knowledge about event consumers
- Mediator: centralized mediation
  - Encapsulates and coordinates communication
- Notification Service
  - Combines the Observer and the Mediator patterns
  - Subscribers only know about events not about publishers
  - Mediation between event producers and consumers

36

Event-based Paradigm

- The (data) producer is the initiator of communication
- Notifications are not addressed to any specific consumer
  - Producers are not aware of consumers
- Consumers issue subscriptions (interests)
  - Consumers are not aware of producers
- Notifications are delivered to consumers if they match with subscriptions
- Flexible!

37

Modes of Interaction

- Influences
  - The architecture of the system
  - The design of the individual processes involved
- Link distributed parts of the system
  - difficult to change afterwards
- MoI determine system's ability to adapt, evolve and scale
- MoI is confused with the implementation techniques

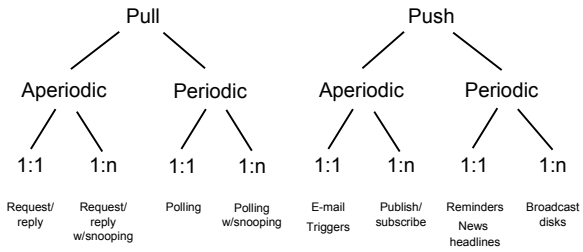
38

Interaction Patterns

- Initiation
  - (client) pull vs. (server) push
  - periodic vs. aperiodic
- Topology
  - 1:1 (unicast) vs. 1:n (multicast)
- Lifecycle
  - time-dependent vs. time-independent
- Concurrency
  - blocking vs. non-blocking
- Reliability
  - atomic, at-least-once, at-most-once, exactly-once

39

Data Delivery Options



40

Interaction vs. Invocation

- Must separate mode of interaction and implementation (invocation) technique
- Separation must occur at various levels of abstraction
  - RPC implemented using messages
  - Implementation using other interaction patterns
    - Pointcast: implemented an event-driven notification service through a polling mechanism

41

Invocation Mechanisms of C/S Sys

- The communication mechanisms used in client/server systems fall into one of the following categories:
  - remote procedure call (RPC)
  - transactional RPC
  - peer-to-peer messaging
  - queues
  - transactional queues
  - events/Publish-Subscribe

42

Middleware

- used to glue together applications (components):
  - IPC by sockets, shared memory
  - TCP/IP, X.25
  - common database
  - RPC, CORBA RMI, J2EE
  - MOM

Message Oriented Middleware

- applications communicate through explicitly sending/receiving messages
- most common flavors:
  - queues
    - point-to-point (mostly)
    - location-based addressing
    - enqueue, dequeue
    - store and forward
  - publish/subscribe
    - different addressing approaches
    - register & callback (Observer pattern)
    - optimize network use

MOM (cont.)

- flexible interaction
  - C/S request/reply, one-way push
  - asynchronous and time-independent
  - 1:1, n:1, 1:n, m:n
  - priorities
- flexible reliability
  - volatile/persistent/transactional queues
  - reliable/certified/transactional pub/sub
- additional services
  - load balancing, naming, security, content transformation

Communication Mechanisms

Already seen Request/Reply

- Simple
  - + imperative nature of C/S paradigm
  - + programming language abstraction
- Drawbacks
  - Point-to-point communication limits scalability
  - Polling limits accuracy of data
    - Unnecessary bandwidth consumption

Queues

- Why use queues?
  - Asynchronous communication
  - No blocking while waiting for reply
  - Clients can submit requests even if server is not available
  - Easy to handle results of disconnected clients
  - Load balancing
  - Possibility of prioritizing the requests in the queue

Operation with Queues

- Persistent queue between client and server
- Client: enqueues requests, dequeues replies
- Server: dequeues a request, processes request, enqueues reply, commits
- If transaction aborts due to system reasons it is enqueued again

49

Queue Managers

- Queue manager needed
  - operations on queue elements: enqueue, dequeue, scan queue, keyed access
  - create and destroy queues
  - modify a queue's attributes, such as owner, size, privileges
  - start and stop queue
  - routing of requests (forwarding to another queue manager in case of overload)

50

Server's View of Queuing

- Assume each request is for execution of a single transaction
- Server dequeues a request, executes the request, enques the result, and commits
- If the transaction aborts
  - the dequeue operation is undone
  - the enqueue operation is undone if already started
- If client checks queues, request is either in request queue, in process, or result in result queue

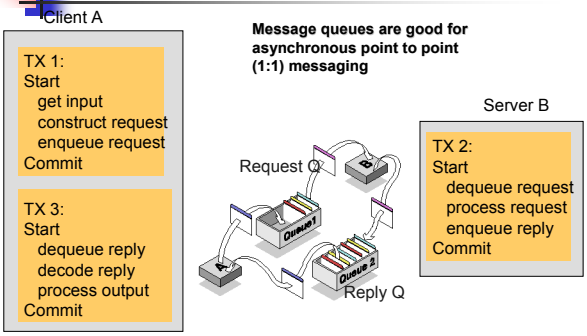
51

Client's View of Queuing

- Client perceives three transactions for each request:
  - one transaction to enqueue request
    - receive input from user, construct request, enqueue request, commit
  - one server transaction (described above)
  - one transaction to dequeue results
    - dequeue reply from result queue, convert to proper output format, deliver output, commit (wiping out result in result queue)

52

Request/Reply with Queues



54

Cost/Benefit of Operating with Queues

- Using queues is expensive
  - 3 transactions instead of one
  - transactional queues must be managed by a (specialized) DBMS to guarantee persistence and transaction semantics
- Using queues buys flexibility
  - communication with unavailable clients or servers
  - load balancing across servers
  - easy implementation of priorities
  - easier integration of legacy systems

55

Need for Persistent Sessions

- System must be able to identify sending and receiving transactions and match them
- Without request/reply semantics, queue manager may not accept requests with output parameters (since results would be simply dumped on device)
- Recovery of queuing systems later (with TPM)

56

Summary communication mechanisms

- RPC: synchronous, simple call-return semantics, hard-wired termination and ordering
- Multicast: 1:N messaging for group communication
- Point-to-point messaging: flexible sequencing of messages, asynchronous (synchronous possible), optimizable, sequencing and timing reflected in application logic therefore more difficult to use
- Queues: fully asynchronous, maximum flexibility for handling client/server/communication failures

57

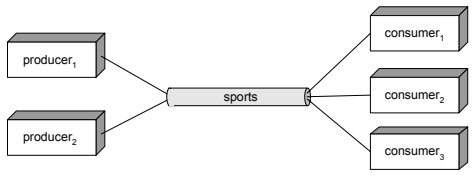
Publish/Subscribe

Pub/Sub Notification Service

- Main characteristics
  - decouples producers and consumers
  - anonymous to each other
  - dynamic number of consumers and producers
  - no directory service is needed
- Addressing models
  - Channel-based
  - Subject-based
  - Content-based
  - Concept-based

59

Channel-based



- Less powerful
- Simple

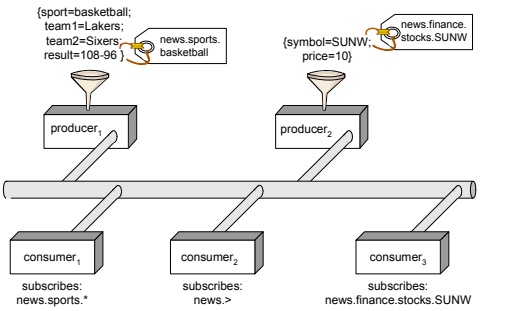
60

Subject-based Addressing

- Subject-based addressing avoids use of physical network addresses
- Senders label a data message with a subject name
  - Subject = characterize/synthesize message content
- Consumers listen to names and pick up the messages with the proper subject name
- Anonymous rendezvous:
  - producers need not know how data is consumed
  - consumers need not know how or where data is produced

61

Subject-based Addressing (2)



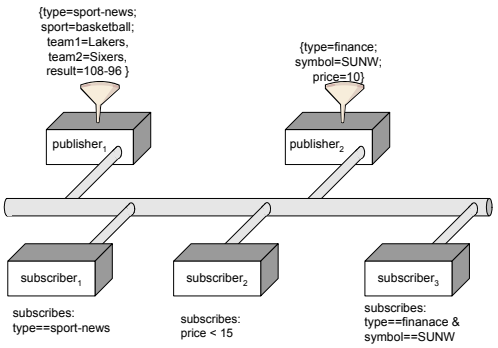
Subject-based Addressing (3)

- Agreement on subject names
- New subjects can be created dynamically
- Subject names consist of elements (subject name hierarchy)
  - element.subelement.subsubelement
- Wildcards can be used
  - RUN.* matches RUN.AWAY
  - RUN.home
  - RUN.> matches RUN.AWAY.far
- Difficult to change subject hierarchy

Content-Based Pub/Sub

- A content-based filter F
  - is a predicate on the content of notifications
  - induces the set of matching notifications
- Content-Based filtering is flexible but complex
  - cannot be easily mapped to "IP-Multicast"
- Centralized implementations not scalable to wide-area scenario
  - powerful distributed infrastructure required

Content-Based Pub/Sub

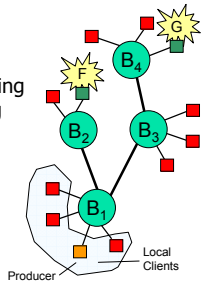


Problems derived from scale

- Flooding of notifications is not an applicable solution
  - need strategies for filter placement to optimize bandwidth and size of routing tables

Content-Based Routing

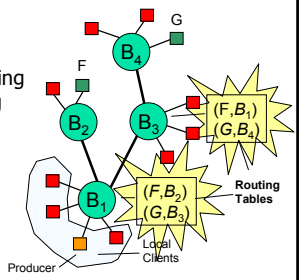
- Cooperating brokers
  - Local clients
  - Notification forwarding
  - Filter-Based Routing Tables
- Tradeoff: Network resource waste vs. filtering overhead (processing and delay)



subscriptions: F and G

Content-Based Routing

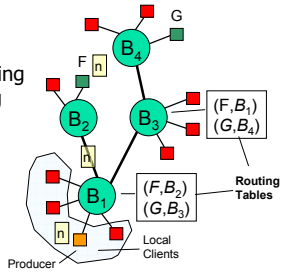
- Cooperating brokers
  - Local clients
  - Notification forwarding
  - Filter-Based Routing Tables
- Tradeoff: Network resource waste vs. filtering overhead (processing and delay)



routing tables are built

Content-Based Routing

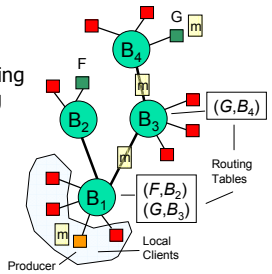
- Cooperating brokers
  - Local clients
  - Notification forwarding
  - Filter-Based Routing Tables
- Tradeoff: Network resource waste vs. filtering overhead (processing and delay)



notification n is published it matches with F

Content-Based Routing

- Cooperating brokers
  - Local clients
  - Notification forwarding
  - Filter-Based Routing Tables
- Tradeoff: Network resource waste vs. filtering overhead (processing and delay)



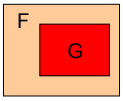
notification m is published it matches with G

Content-Based Routing (cont)

- Size of routing tables crucial for scalability
  - global knowledge about all active subscriptions not feasible
- Solution: reduce size of routing tables and overhead to update them by
  - exploiting similarities among filters
    - identity tests
    - covering tests
  - merging of filters
  - trading accuracy vs. efficiency

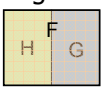
Covering

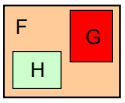
- Filters can cover each other
- Covering can decrease
  - size of routing tables
  - filter forwarding overhead



Merging of Filters

- Filters can be merged
  - *perfect*
  - *Imperfect*





- Merging generates new covers
  - similar benefits as covering

The REBECA Approach

- Prototype of notification infrastructure
  (**R**EBECA **E**vent-**B**ased **E**lectronic **C**ommerce **A**rchitecture)
  (http://www.gkec.informatik.tu-darmstadt.de/rebeca)
- Content-based routing with optimizations
  - Flexible filter framework
  - Support for complex data types
- Structuring publish/subscribe systems
  - Scoping
  - Sessions

74

Concept-based Pub/Sub

- Main advantages of Publish/subscribe
  - decouples producers and consumers
  - anonymous to each other
- BUT even though consumer and producer use a common vocabulary (let's suppose this) assumptions of participants are implicit
  - (date) 7/11/2003 Which is the month?
  - (price) 200 Currency? €?, U$S?…
- Subscriptions expressed on flat messages

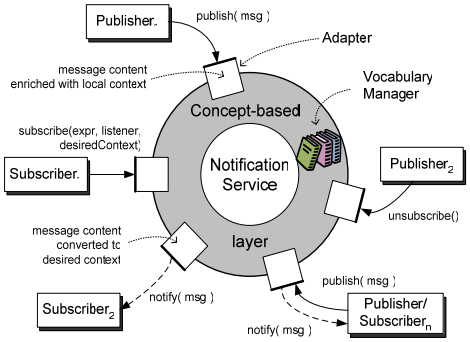
75

Concept-based Pub/Sub

- Provide a higher level of abstraction to describe the interests of publishers and subscribers
- Events represented using MIX
- Subscribers can specify their assumptions
  - Price < 100 [€]
  - DeliveryDate <= 7/11/2003 [dd/mm/yyyy]
- The notification service delivers ready-to-process events to subscribers
  - No further processing is needed

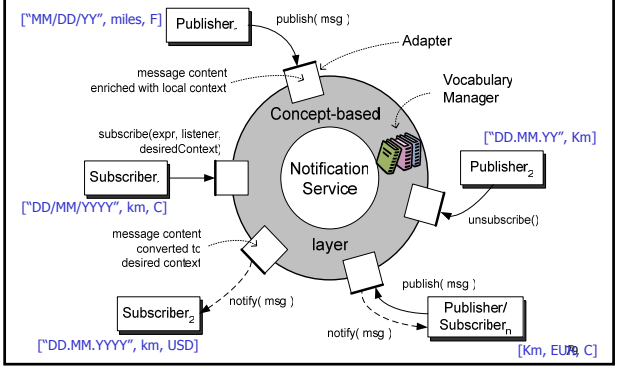
76

Concept-based Overview



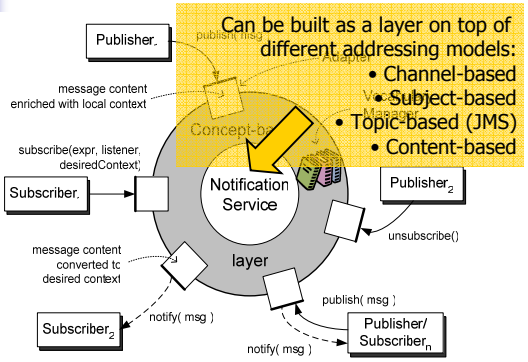
78

Concept-based Overview



79

Concept-based Overview



80

Wrap up

- Different routing strategies according to application needs
- Filters/subscriptions on a single message
- Event correlation no supported
  - Need to cache/store semi-composed events
- Software Engineering
  - Need to scope events and subdivide event space

81

Data Dissemination Products

Queue Managers: IBM's MQSeries

- most TP Monitors offer queue managers (TUXEDO, Encina, TOP END)
- standalone products (IBM's MQSeries, BEA messageQ, SonicMQ, SUN JMQ, ...)
- MQSeries provides interoperable queue management across many Operating Systems (~15)
- works with all IBM TP monitors and any system supporting the X/Open XA interface (including CORBA OTS), Java connectivity included
- when working with a TPM, MQSeries uses the TPM transactions, otherwise it provides its own

91

MQSeries (cont.)

- multiple named queues supported
- queue forwarding among queues (e.g. for load balancing)
- queue forwarding occurs within channel agent's own transaction
- pub/sub brokering possible
- queue manager consists of
  - connection manager
  - data manager
  - lock manager
  - buffer manager
  - recovery manager
  - log manager

92

MQSeries: Qs, API ...

- types of Qs
  - local (app., transmission, dead-letter, initiation, ...)
  - remote, alias, model, dynamic
- interaction through MQI verbs
  - MQBEGIN, MQCMIT
  - MQPUT, MQGET (browsing, consuming, blocking/non-blocking)
  - control operations
    - connect/disconnect Qmanager (MQCONN, MQDISC)
    - set configurations, manage Q processing (MQOPEN, MQSET, MQCLOSE)
- interaction through C++/ Java APIs
- interaction through JMS API

93

MQSeries Messages

- messages can be
  - persistent
    - more secure, more expensive, logged, exactly once semantic
  - non-persistent
    - less secure, faster since in main memory, at most once semantic
- both types of messages can be enqueued in same queue
- message data
  - user defined format
  - default format and encodings

94

MQSeries: Messages (cont.)

- message consists of descriptor and data
- descriptor includes context
  - identity
  - origin
  - system message ID
  - application message ID
  - message type
    - datagram, request, reply,
    - report
- persistence flag
- name of destination queue
- ID of reply queue
- correlation ID
- expiry
- application-defined format
- report options
  - confirm on arrival, on delivery, on positive/negative action, on expiration, or on exception
- priority

95

MQSeries (cont.)

- management of message processing applications
  - process definition associated with Q
  - Qmanager sends trigger to *initiation* Q
  - trigger monitor may start application using process definition in trigger message

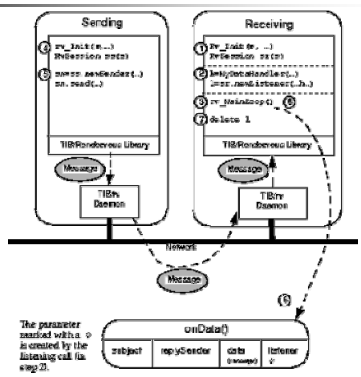
96

TIBCO's TIB/Rendezvous

- Event-driven, publish/subscribe
- Subject-based addressing
- Self-describing messages
- Leverage on broadcast & IP-multicast

97

TIB/Rendezvous Architecture



98

TIB/Rendezvous Messages

- Data Messages are self-describing
  - data + descriptive information
    - data
    - length of data
    - datatype indicator
    - subject name
  - listener callback functions receive same bundle
  - automatic conversion between local data format and TIB/Rendezvous wire format

99

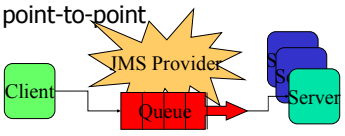
Java Message Service - JMS

- Transactional, asynchronous messaging
- De-facto standard for Java messaging APIs
- Supported by almost every QueueManager vendor (IBM, Oracle, BEA,...)
- Many 100% Java, lightweight JMS products (Fiorano, Progress, Softwired, SpiritSoft, etc.)
- Designed for portability
  - Interfaces only => many different realizations
  - APIs to create, send, receive, read messages

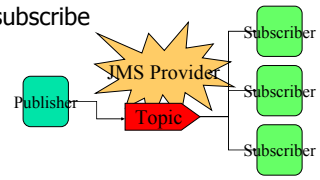
104

JMS Model

- Supports both models
  - Queues for point-to-point



  - Publish/subscribe



105

JMS Model

- JMS point-to-point
  - Queue object: encapsulates provider specific Q name
  - QueueConnection: handle to underlying transport
  - QueueSession: produces and consumes messages
  - TemporaryQueue: temporary storage for the QueueConnection
  - QueueConnectionFactory: creates QueueConnection
  - QueueReceiver: gets messages
  - QueueSender: puts messages

106

JMS pub/sub

- JMS publish/subscribe:
  - Combination of
    - channels (now topics) and
    - expressions on envelope's attributes
  - Factories, destinations, etc. identified via JNDI

107

JMS Messages

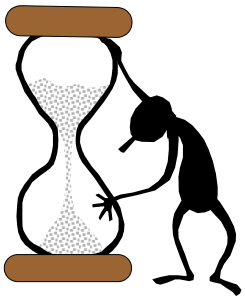
- Message types
  - text
  - map: (name,value) pairs
  - object: serializable object
  - stream: primitives
  - byte
- Message header used for addressing
- Message properties
  - list of (name,value) pairs
  - selectors: SQL-like conditional expressions, MyType='car' AND MyName like 'Mu%'

108

JMS Open Issues

- load balancing
- scalability/availability
- fault tolerance
- error notification
- end-to-end security
- segregation of domains
- simple and flexible deployment configuration
- Many of these issues being addressed by vendors

110



112