Controlling Historical Information Dissemination in Publish/Subscribe

Jatinder Singh, David Eyers and Jean Bacon

Opera Group
Computer Laboratory
University of Cambridge
Outline

Motivation
- Overview of event-based middleware
- Need for replay mechanisms

Data control model

Event replay mechanisms
- Unified approach for delivery of past/future events
- Subject to similar disclosure mechanisms

Application scenarios
- Health domain
Motivation

overview of publish/subscribe and event replay
Publish/Subscribe

Event-based middleware
- Events - Data-rich encapsulation with a particular semantic
- Represents a incident, as it occurs
  - E.g. emergency detected
- Middleware – layer of indirection between clients and network

Publish/Subscribe
- Publishers produce information (events)
- Subscribers receive events of interest
- Brokers cooperate to route information on type/content

Features
- Scalability, load balancing
- Decoupling producers/consumers
Historical Information - Event Replay

Queries
- Past (query) vs future (subscription)

Q: Where to query?
- Pub/sub is decoupled
- Information comes from various sources
- E.g. Contract nurse treated patients from over 50 institutions – what controlled drugs did she authorise?

Circumstances are important
- Occurrences make previous information relevant
- Replay (can be) more than just a snapshot
- Related work – replay buffers

Security aspect
- Situations authorising replay
- Data controls mechanisms differ according to context

Historical dissemination mechanisms need control
Data Control Model

controlling data flow through brokers
Interaction Control

Customising data disclosure to circumstance
- Tailoring data to the situation
  - Need-to-know

Control data released by a pub/sub broker
- Administrators define policy to meet responsibilities

Context-sensitive policy rules
Restrictions

Subscription authorisation
- Authorise event channel
  A **DOCTOR** can subscribe to a **TREATMENT** event where `treats(nhs_id, sub.patient_id)`

Event restrictions
- Restrict an event instance (message specific)
- Imposed silently
  **Dr X** only receives **TREATMENT** events for **Jill** where `ev.condition≠HIV`

Sensitive to context
- Restrictions reference relationships, qualifications, etc.
- Reactive to change
Transformations

Customising an event to circumstance

Transformation functions

- Altering attribute values of the instance
  - Change granularity, remove/mask
- Produce another event type
  - Enrich, degrade or unrelated event
  - May consume the original

Apply on publication or notification

*More than binary access control*....
Context

Rules are defined with reference to context

Database integration
  - Unifying storage/transmission

Rich representation of state
  - Messaging information (event instance)
  - Credentials of the principals (e.g. Dr, Surgeon, Nurse)
  - Stored data, functions
    - Conditions, fluent state, etc.

Access mechanisms are context sensitive
Controlling historical data dissemination

extension for event replay
Requests – Subscriber Initiated

Replay request

```xml
<replay_request>
  <event_type />
  <filter />
  <from />
  <to />
  <during />
</replay_request>
```

Subscription request

- Query over *future* data

=> Unification of subscription/replay requests

  - Through addition of a `<FROM />` tag
Authorising Replays

Defines a valid replay request

ALLOW REPLAY OF (event type)
WHERE (validation conditions);

ALLOW REPLAY OF prescribe
WHERE investigating(req.prescriber_id)
AND credentials(user, auditor);
Server Initiated Replay

Policy might automatically replay historical events
- Quality of service, change in context affects access control
- Delivered message includes replay metadata

Facilitated through auto-replay rules

AUTO REPLAY EVENT TYPE (type)
(FROM | TO) (timestamp)
((NOT) DURING) (fluent)
WHERE (validation conditions)
FILTER (filter conditions)
ON EVENT (type) where (trigger conditions)
Replayed events are subject to the same interaction control mechanisms

Authorisations define general privilege (can sub/replay)
  – Qualified by restriction/transform rules
    • Separation brings flexibility – many-to-many mapping

Changes in context affect data distribution
  – Interaction control mechanisms applicable at publication time may differ to those at replay time
Scenarios

application to healthcare scenarios
Drug monitoring

Controlled drug legislation:
- Nurses can prescribe drugs
  - May be controlled (e.g. morphine) in certain cases
- Controlled drugs must be monitored
  - Audit is prescriber focused
    - Patient privacy should be protected

Data control scenario
- Nurse publishes \texttt{preserve} events
- Auditor needs information on controlled drugs
- NO patient specifics, except where the prescriber is under investigation
  - Replay previous \texttt{preserve}
Drug Monitoring

<replay_request>
  <event_type>prescribe</event_type>
  <during>not investigation(nurse_8821)</during>
  <filter>prescriber_id = nurse_8821</filter>
</replay_request>

RESTRICT DELIVERY OF prescribe
WHERE credentials(user,auditor)
FILTER controlledDrug(prescribe.drug_id);

TRANSFORM EVENT prescribe TO prescribe
ON NOTIFY (auditor)
EXECUTE perturb_for_auditor()
WHERE not investigating(prescribe.prescriber_id);

ALLOW REPLAY OF prescribe
WHERE not req.prescriber_id is NULL
    AND credentials(user,auditor);
Sensor monitoring

Healthcare shift to homecare provisioning

Home scenarios use sensors
- Monitoring various aspects of physical state
  - ECG, heart rate, respiration rate, temperature, movement, posture, positioning
  - Location: room, GPS coordinates

Privacy
- Perturb some information from the stream
  - E.g. Summary values (OK/60-80); HOME/NOT HOME
- Only in non-emergency situations
- At patient request

Auto-Event Replay
- Replay detailed sensor readings until the emergency
Sensor monitoring

```sql
TRANSFORM EVENT vital_signs TO vital_signs
ON publish EXECUTE perturb_readings()
WHERE not emergency(vital_signs.patient_id);

AUTO REPLAY EVENT TYPE vital_signs
FROM now() – interval '2 hours' TO now()
WHERE sub.patient_id = nhs_pat_4122
FILTER vital_signs.patient_id = nhs_pat_4122
ON emergency
    WHERE emergency.patient_id = nhs_pat_4122;
```
Conclusion

Event replay is useful, particularly where
- Changes in context alter visibility
- Information comes from multiple sources

Considered data access (security)
- Future work: rate/flow control;
  - Facilitated through existing mechanisms?

Data control framework unifying the delivery of past and current/future events
- Allows fine-grained control over information release
  - Administrator meets responsibilities
  - Simplifies client access
  - *Active log/audit*
Interaction Control Policies

Control data released by a broker

Context-sensitive policy rules
  - Messaging information (event instance, principal)
  - Credentials of the principals (e.g. Dr, Surgeon, Nurse)
  - Environmental state
    - Stored data, external functions

Enforced in local brokers
  - Reactive to events
    - At stages of the messaging process
  - Produce and consume events

Database Publish/Subscribe
  - Brings rich representation of state & data handling capabilities to the messaging system
Application to Healthcare

Middleware
- Suits data-driven environments
- Favours interoperability

Broker-level data control
- Lightweight clients
- Ensures policy adherence
- Suits federated administrative environments

Gives a domain control over data they manage
- Controlled by administrative domains (service providers)
  - Ground principals, hold/store/share information
- In line with responsibility
- Aids in accountability
  - Depends on design (centralised vs. federation)
Database Publish/Subscribe

Enterprise applications concern data
Build publish/subscribe substrate into DB
- Common type interface
- Replication
- Transactional semantics
- Relational model, query languages
- Performance, maintenance, etc...

Broker = Pub/Sub DB instance
- Routes messages
- Stores/consumes data
- Act as a publisher and/or subscriber
  - Produce & consume events

Rich representations of context