Expectations and Reality in Large-Scale, Widely Distributed Systems

Jean Bacon

University of Cambridge Computer Laboratory
Outline

• problems and some thoughts on why we have them
• what’s solved? what’s hard? what’s new?
• categories of large-scale distributed system
• promising approaches
• research experience at Cambridge
• still to be solved?

Work-in-Progress highlighted throughout
Costly Failures - 1

- **UK Stock Exchange** - share trading system
  - abandoned 1993, cost £400M
- **CA automated childcare support**
  - pended 1997, cost $300M
- **US tax system modernisation**
  - scrapped 1997, cost $4B
- **UK ASSIST**, statistics on welfare benefits
  - terminated 1994, cost £3.5M
Costly Failures - 2

- London Ambulance Service Computer Aided Despatching (LASCAD)

scrapped 1992, cost £7.5M, 20 lives lost in 2 days,
tracking of all ambulances, GIS, automatic allocation,
event-driven, rule-based approach
  - unrealistic schedule
  - lowest bidder selected, had no experience
  - backup system not checked
  - no testing/overlap with old system
  - users not consulted during design, lacked confidence
  - simple programming error: storage not deallocated
Why high public expectation?

Web experience

e.g. general information services
e.g. online banking
e.g. airline reservation
e.g. conference management
e.g. online shopping and auction

Properties: read mostly, server model, client-server paradigm, closely coupled, synchronous interaction, single-purpose, private sector
Public Sector Systems
healthcare, police, social services, immigration, passports, vehicle-drivers licensing

• large scale
• bespoke and complex
• many types of client (many roles)
• web portal interface, but not weblike service model
• long timescale, high cost
• legislation and government policy
Some Legal/Policy Requirements - 1

“patients may specify who may see, and not see, their electronic health records (EHRs)”

“only the doctor with whom the patient is registered (for treatment) may e.g. prescribe drugs, read the patients EHR, etc.”

“the existence of certain sensitive components of EHRs must be invisible, except to explicitly authorised roles”
Some Legal/Policy Requirements - 2

“buses should run to time and bus operators will be punished if published timetables are not met.”

so bus operators refuse to cooperate in traffic monitoring, even though monitoring could show that delay is often not their fault.
Data Protection Legislation

Gathered data that identifies individuals must not be stored:

**CCTV cameras**: software must not *recognise* people and store identities with images

**Vehicle number plate recognition**: must not be associated with people then stored with identities

**Police records**: accusations that are not upheld? *(e.g. Soham murders)*

**UK Freedom of Information Act**: Jan 2005, should we design with disclosure in mind?
Rapid Development of Technology

• Can’t ever design a “second system”, it’s always possible to do more next time

• Rapid obsolescence - incremental growth is difficult

• But big-bang deployment is a bad idea
design for incremental deployment
New technologies to incorporate

- Mobile workers in healthcare, police, utilities etc.
  Integration of wired and wireless networks

- Integration of camera and sensor data
Outline

• problems and some thoughts on why we have them
• what’s solved? what’s hard? what’s new?
• categories of large-scale distributed system
• promising approaches
• research experience at Cambridge
• still to be solved?

Work-in-Progress highlighted throughout
Structures for Large-Scale Systems

1. Federated administration domains
   – integration of databases
   – integration of sensor networks

2. Independent, external services

3. Detached, ad-hoc groups
1. Federated administration domains

- **security**: firewall-protected
- **names** administered (services, principals, roles, ….)
- **policies** specified e.g. for authorisation, plus some external policies to satisfy government policy, legal and institutional requirements
- **high familiarity, high trust**

The second half of this talk relates to this structure
Examples

• national healthcare services:
  many hospitals, clinics, primary care practices.
  external services – e.g. national EHR

• national police services:
  52 county police forces,
  external services e.g. DVLA, court-case workflow

• global company:
  branches in London, Tokyo, New York, Berlin, Paris ..

• active city:
  fire, police, ambulance, healthcare services.
  mobile workers
  sensor networks e.g. for traffic monitoring
2. Independent, external services

- naming and authentication
  client-domain-related and/or of individuals via certification authorities
- authorisation policies
  related to client roles and/or individual principals
- need for: charging, accounting, audit
  a basis for mutual trust (service done, client paid)
- trust
  based on evidence of behaviour,
  clients exchange experiences, services monitor and record
  assume full connectivity, e.g. with CAs, so can authenticate/identify

Examples: e-science (grid) services, for computation (e.g. XenoServices)
  and databases (e.g. astronomical, medical, transport)
3. Detached, ad-hoc groups

• e.g. connected by wireless
• can’t assume trusted third-parties (CAs) accessible
• can’t assume knowledge of names and roles, identity likely to be by key/pseudonym
• new identities can be generated (by detected villains)

• parties need to decide whether to interact
• each has a trust policy and a trust engine
• each computes whether to proceed – policy is based on:
  - accumulated trust information
    (from recommendations and evidence from monitoring)
  - risk (resource-cost) and likelihood of possible outcomes
Simplified SECURE Trust Model

- trust formation
- evidence manager
- interaction monitor
- trust calculator
- risk evaluator
- entity recognition
- access control
- request (p, ..)

Observations

decision

request (p, ..)
Outline

• problems and some thoughts on why we have them
• what’s solved? what’s hard ? what’s new?
• categories of large-scale distributed system
• promising approaches
• research experience at Cambridge
• still to be solved?

Work-in-Progress highlighted throughout
Promising Approaches for Large-Scale Systems

- **Roles** for scalability
- **Parametrised roles** for expressiveness
- **RBAC** for services, service-managed objects, including the communication service
- **Policy** specification and change management
- **Policy-driven** system management

- **Asynchronous**, loosely-coupled communication publish/subscribe for scalability
  - **Event-driven paradigm** for ubiquitous computing
- **Database** integration – how best to achieve it?

And don’t forget:
- **Mobile** users
- **Sensor network** integration
Opera Group – research themes
(objects policy events roles access control)

• Access Control (OASIS RBAC)
  Open Architecture for Securely Interworking Services
• Policy expression and management
• Event-driven systems (CEA, Hermes)
  EDSAC21: event-driven, secure application control for the 21st Century
• Trust and risk in global computing (EU SECURE)
• TIME: Traffic Information Monitoring Environment

see: www.cl.cam.ac.uk/Research/SRG/opera

for people, projects, publications for download
Access Control
Motivating example: a national Electronic Health Record (EHR) service. Police and Social Services are similar

- MUST protect EHRs from journalists, insurance companies, family members etc.
- access policy defined both nationally and locally
- generic scalable policy \( \Rightarrow \) RBAC
- exception of individuals is allowed by law, (all doctors except my uncle Fred Smith)
  “Patients’ Charter” \( \Rightarrow \) parametrised roles
- may need to express relationships between parameters
  
  \textit{treating-doctor ( doctor-id, patient-id )}
Access Control: Requirements / Motivation

- large scale
  => role based access control (RBAC)
- potentially widely distributed systems
- heterogeneous components, developed independently but must interoperate
  => service-level policy agreements (SLAs)
  (which roles authorise their activators to use which services?) negotiated within and between domains
- incremental deployment
OASIS RBAC

- OASIS services name their clients in terms of roles

- OASIS services specify policy in terms of roles
  - for role entry (activation)
  - for service invocation (authorisation, access control)
    both in Horn clause form
OASIS model of **role activation**

a role activation rule is of the form:

```
condition1, condition2, ..... |- target role
```

where the conditions can be

- prerequisite role
- appointment credential
- environmental constraint

all are parametrised
OASIS role (continued) membership rules

as we have seen, a role activation rule:

\[
\text{cond1}^*, \text{cond2}, \text{cond3}^*, \ldots \vdash \text{target role}
\]

role membership rule:

the role activation conditions that must remain true, e.g. *
for the principal to remain active in the role

monitored using event-based middleware
another contributor to an active security environment
OASIS model of authorisation

An authorisation rule is of the form:

\[
\text{condition}_1, \text{condition}_2, \ldots \vdash \text{access}
\]

where the conditions can be

- an active role
- an environmental constraint

all are parametrised
A Service Secured by OASIS Access Control

- **principal**
- **credentials**
- **role entry**
- **policy**
- **credential records (active roles’ status)**
- **access control**
- **OASIS-secured service**

- **RMC = role membership certificate**
- \(\text{RMC} = \text{role entry} \quad \text{or} \quad \text{RMC} = \text{use of service}\)**

**Check persistent credentials and environmental constraints**
**Check environmental constraints**

**monitoring heartbeats or change events**
Active Security Environment
Monitoring membership rules of active roles

- **RMC** = role membership certificate
- **CR** = credential record
- **ECR** = external credential record

---

**a prerequisite role for service C’s role**

**service A**

**service B**

**service C**

*heartbeats or status-change events*
Event-based Systems

Asynchronous Communication
Event-Driven Systems (1)

Cambridge Event Architecture (CEA), 1995 -

• extension of O-O middleware, typed events
• federated event systems:
  gateways/contracts/XML
• applications:
  multimedia presentation control, pervasive environments
  (active house, active city, active office),
  tracking mobile entities (active badge technology),
  telecommunications monitoring and control
Event-Driven Systems (2)

**Hermes** event service, 2001- 4

work of Peter Pietzuch

- loosely-coupled, **publish/subscribe**
- widely distributed event-broker network
- over a P2P overlay network
- distributed filtering (optimise use of comms.)
- rendezvous nodes for advertisers/subscribers
Hermes Pub/Sub Design

- **Event Brokers**
  - provide middleware functionality
  - logical overlay P2P network with content-based routing and filtering
  - easily extensible

- **Event Clients** (Event Publishers, Event Subscribers)
  - connect to any Event Broker
    - publishers *advertise,*
    - subscribers *subscribe* (brokers set up routing state),
    - publishers *publish,*
    - brokers route messages and *notify* publications to subscribers
  - lightweight, language-independent
Algorithms I – Topic-Based Pub/Sub

- Type Msg, Advertisements, Subscriptions, Notifications
- Rendezvous Nodes
- Reverse Path Forwarding
  - Notifications follow Advs and then the reverse path of Subs
Algorithms II – Content-Based Pub/Sub

- Filtering State
- Notifications follow reverse paths of subscriptions
- Covering and Merging supported
Implementation

• **Actual Implementation**
  – Java Implementation of Event Broker and Event Clients
  – Event Types defined in XML Schema
  – Java Language Binding for Events using Reflection

• **Implementation within a Simulator**
  – Large-Scale, Internet-Like Topologies
  – up to $10^4$ Nodes so far
But pub/sub is not sufficient for general applications

- decouples publishers and subscribers
  *pubs/subs need not be running at the same time*
- publishers are anonymous to subscribers
  *subs need to know topic(attributes), not pubs’ names and locations*
  *but receivers may need to know the sender or sender’s role*
- only multicast, one-to-many communication
  *may also need one-to-one*
- can’t reply
  *either anonymously, e.g. to vote, or identified*
- efficient notification for large-scale systems
  *but one-to-one should also be efficient – optimise*

*Work-in-Progress to generalise Hermes*
Event-driven systems (3)

Event composition (correlation)

Pietzuch, Shand, Bacon, Middleware 2003,
IEEE Network, Jan/Feb 2004

• composite event service above event brokers
• service instances placed to optimise communication
• FSM recognisers – parallel evaluation
• events have source-specific interval timestamps
• simulations of large-scale systems …in progress
Bottom-up and/or Top-Down?

• Can we express all we require by bottom-up composition of primitive events?
• Do we also need high-level models of context?  
  e.g. maps, plans, mathematical models  - YES
• What can users be expected to express?
• How is the top-down, bottom-up gap bridged and high-level requirements converted into event subscriptions?

Work-in-Progress
Integrating sensor networks (1)

Application

Event Communication and Composition

aggregation, inference, storage, control

sensor clusters

Context models

device control

devices

Event Databases

Device control

devices

event flow

control flow
Integrating sensor networks (2)

• heterogeneous sensors abstracted into events
• inaccuracies masked – data cleansing?
• value aggregation?
• timestamping?
• real-time delivery guaranteed?

e.g. traffic monitoring via IR, acoustics, counting applications subscribe to:

“car-event (...), “bus-event (#4, ..... )”, “taxi-event (.....)”

Work-in-Progress
Integrating databases with pub/sub

• note: continuous queries require recording of individual queries and individual response, one-to-one.
• instead: databases advertise events:
  *event type (\texttt{<attribute-type>})* based on virtual relations
• clients subscribe and are notified of occurrences
• we use PostgreSQL - active predicate store

Work-in-Progress
Motivating Example – Police IT

- *Fred Smith is suspected of masterminding a nationwide terrorist organisation.*

- As well as looking up his past database records, the investigators subscribe, in all counties, to advertised database events specifying his name as an attribute.

- Triggers are set in the databases so that any future entries that are made, relating to his movements and activities, will be notified automatically and immediately to those investigating him.
Securing pub/sub using RBAC

At the event client level – use RBAC
• domain-level authorisation policy indicates, for event types and attributes, the roles that can advertise/publish and subscribe
• inter-domain subscription is negotiated, as for any other service
• note that spamming is prevented – only authenticated roles can use the pub/sub service to advertise/publish

At the event-broker level – use encryption
• are all the event brokers trusted?
  if not, some may not be allowed to see (decrypt) some (attributes of) some messages.
  this affects content-based routing.

Work-in-Progress
Outline

• problems and some thoughts on why we have them
• what’s solved? what’s hard? what’s new?
• categories of large-scale distributed system
• promising approaches
• research experience at Cambridge
• still to be solved?

Work-in-Progress highlighted throughout