EXPECTATIONS AND REALITY IN DISTRIBUTED SYSTEMS

Jean Bacon
Computer Laboratory, University of Cambridge
email:Jean.Bacon@cl.cam.ac.uk

ABSTRACT
There is widespread awareness of the potential of network-based systems. Governments expect to be able to procure nationwide distributed systems for healthcare, taxation, policing, social services, transport. Yet even centralised large-scale systems come in over-time, over-budget or never complete.

The state of the art, in current practice and standards, is not yet able to provide fully general and efficient global communication, with integrated database services. Nor are current systems able to enforce the fine-grained policies required by law. A consensus is emerging that asynchronous, publish/subscribe-based communication and role-based access control are appropriate for large-scale distributed systems. But further research in these areas is needed before the requirements for tomorrow's distributed systems can be met. In addition, new challenges are constantly being presented by the increasing use of mobile devices and sensor technology. Some research directions to meet these requirements are presented.

KEY WORDS
publish-subscribe, role-based access control, policy, trust

1 Preamble

The unprecedented growth in computer and communications technology is set to continue. The challenge for computer scientists is to manage the complexity of the systems such technology makes possible. The successful complex systems of the past had time to evolve; to start small and grow by small increments, deploying new technology as it became available. An example is the telephone service. The rapid evolution of computer technology, and its ubiquitous deployment, create a situation beyond our experience.

There is widespread public awareness of computer networking because of the Internet-based world-wide web and the multitude of services that are now provided via web browsers, irrespective of national boundaries. This awareness profoundly influences the expectations of citizens and those who make policy on behalf of government and business. It is assumed that large-scale, network-based distributed systems can be designed and deployed.

Even centralized systems projects can fail: many large-scale, centralized implementations have failed or have been late and massively over-budget (those managing passports, vehicle and driver registration, immigration, taxation, traffic light control, ambulance scheduling). The likelihood of failure of a distributed implementation is much greater because of increased complexity and the many additional factors that have to be taken into account. These include naming, access control, secure communication, failures of components and connections, heterogeneity and high latency. But, based on world-wide web experience, people expect distributed systems to be provided.

There is a danger that many such systems will be proposed and put out to tender and that commercial companies, in order to gain contracts, will give the impression that it is well known how to build them. Experience from the failed London Ambulance Service system is that the lowest bid can win, irrespective of previous experience and track record, and the resulting system can cost lives.

Computer science research has made good contributions in areas that are components of complex, large-scale, distributed systems. We know how to manage naming for large-scale systems and how to perform name to location mapping. We know we have to replicate services to achieve "24 by 7" availability and we know some of the trade-offs between achieving strong and weak consistency of the replicated data. We can select a means of securing a communications infrastructure. In many of these areas research has fed through into standards.

But current practice does not fully meet the requirements of many large-scale distributed systems. Policy-mechanism separation is a well-known guideline for system structuring, in the sense of policy as part of the computational framework. For many large-scale systems, high level policy must reflect government edict and legislation. These may specify requirements that are necessary socially and politically, but that cannot easily be implemented. Many of them concern exceptions to general rules, relationships between principals, and privacy. Examples from the healthcare domain are [21]: "doctors may see the healthcare records only of their registered patients" " the patient can indicate that certain principals may not see their healthcare record", "the fact that certain components of healthcare records exist must be invisible to some principals". Research is needed into how best to meet such essential requirements.

In some cases, government may specify what seems to be desirable policy at a high level without foreseeing the consequences. For example "buses should run to time and bus operators will be punished if timetables are not met". This particular piece of policy means that bus companies refuse to have their buses monitored, even though the monitoring could show that their lateness is often due to circumstances beyond their control. Reliable arrival-time information would make the public more likely to use buses, but is prevented by lack of monitoring. This is at a time when government is trying to increase the use of public transport to reduce overall vehicle emissions and congestion in cities. Ideally, this type of policy should be identified and avoided.

New challenges arise as technology evolves. The software developed to support, for example, healthcare workers and police, must increasingly take into account the fact that they are mobile. Also, interaction via wired and wireless networks with small devices at changing locations must be supported. Another change is the widespread deployment of cameras and sensors. Low-level sensor data should be cleaned, aggregated and presented to applications in a meaningful form.

2 Domain - Structured Global Computing

A prominent style of distributed system comprises a number of independently administered, firewall-protected, named domains. Each domain provides a context for registering names (of principals, roles, services), expressing and enforcing security policy, managing resources etc. Examples are:

(1) A global company has branches in California, London, Paris and Tokyo. While much communication and data is branch/domain-local, some data and events should be shared between branches.

(2) A national police IT system comprises a number of county-level police domains. The domains will make use of external database services such as national vehicle registration and court-case tracking, typically web-accessible. While some data and communication remains county/domain-local, there is a need for inter-domain communication of emergency, traffic and crime events. Also, privacy legislation must be honoured.

(3) A national health service comprises a large number of independent hospitals, clinics, primary-care practices etc. A national Electronic Health Record (EHR) service is to be accessible from all domains.

(4) A city has many independent public services such as police, fire, ambulance, hospital, utilities. As well as communicating with similar services nationally (e.g. police with police) the different services need to interoperate, especially in emergencies.

(5) A city is to be made sensor-rich for the purpose of monitoring traffic. Many potential service domains, relating to different forms of transport such as buses, taxis and private cars, require specific information from the sensor networks.

The domains may be related, such as healthcare institutions, departments within a university or county police services, and will share a communication infrastructure. Any domain may have a parent domain, such as a national police or national health service, where some high level policy relating to all domains may originate. Each domain may, in addition, define many local policies.

Key requirements of these large-scale systems are controlled use of an efficient, expressive, shared communication infrastructure and controlled access to data. The latter may be of a personal nature and its storage and communication must be in accordance with privacy legislation. Two emerging paradigms that promise to be able to meet these requirements are asynchronous, publish/subscribe-based communication and role-based, policy-driven access control for system management [5].

The concept of role is well established for providing scalability. RBAC [15, 28] separates the administration of people, and their association with roles, from the control of privileges for the use of services (including access to service-managed data). Service developers need only be concerned with specifying access policy in terms of roles, and not with individual users. Authentication into roles must, of course, be securely enforced.

We envisage that independently administered, related domains will define similar roles, for example within police domains, healthcare and other public services as well as within branches of a global company. However, current RBAC standards [1] are insufficient to support the fine-grained policy requirements of many systems [21]. In Section 3 we outline our work on parametrised RBAC: OASIS (Open Architecture for Securely Interworking Services), designed to meet these shortcomings.

In most cases, the bulk of the communication is likely to be intra-domain, but there is also a clear need for inter-domain communication and use of external services. Defining communication in terms of roles obviates the need to know the names of individuals carrying out organisational tasks, and when people start, change and leave jobs in distant domains.

Publish/subscribe [14] allows loose coupling between mutually anonymous components and supports many-to-many communication. Loose coupling is essential when application components may be widely distributed and may reside on a range of devices from mobile phones to clustered super-computers, and when their connectivity may be transient. But publish/subscribe currently supports only multicast communication between anonymous parties; the ability for mutually known parties to communicate one-to-one has been lost. In Section 4 we discuss how the multicast communication patterns might be extended while retaining the efficiency of publish/subscribe for large-scale systems. Our EDSAC21 project (Event-Driven, Secure Application Control for the 21st century) aims to meet this need. We also motivate the requirement for databases to be made active and to be integrated with the communication service, another aspect of our EDSAC21 work.

RBAC is capable of securing any service, including the communication service. RBAC policy within a domain may specify which roles may advertise, publish and subscribe to which events [7]. We offer fine-grained security, in that message attributes are encrypted selectively, with key management transparent to the client domain, see Section 5. Negotiated, service-level policy agreements between domains are set up to control inter-domain communication.

In future, commercially-provided communications services may emerge for unrelated domains to share. Clients will then need strong assurances of security and privacy to have sufficient confidence to use the service. The fact that advertisement is required before messages can be published prevents the spam that pervades email communication between humans. Also, subscription before delivery prevents unwanted messages. Without such control unwanted communications could degrade large-scale inter-software communication in the same way that it consumes resources in email management. In our EDSAC21 system a spammer could only be an authorised, authenticated member of a role and therefore can be held accountable.

A multi-domain architecture allows the familiar, secure operation within an administration domain to be extended for global computing. Inter-domain interaction is strictly controlled by negotiated policy [9]. As small devices become increasingly prevalent, connected via MANETs (Mobile, Ad-hoc Networks), a less structured model of global computing emerges. Here, mobile users may be disconnected from their home domains, may encounter other users and services and may wish to interwork with previously unknown entities. There is no longer a substructure to support naming, authentication, access control etc. In such a scenario the notion of trust becomes significant. In Section 6 we outline how trust might be computed, work done as part of the EU SECURE project (Secure Collaboration among Ubiquitous Roaming Entities). Section 7 offers some conclusions and suggestions for future work.

3 Role - Based Access Control

Role-Based Access Control (RBAC) [28] is an established technique for simplifying scalable security administration by introducing *roles* as an indirection between *principals* and *privileges*. Privileges, such as the right to use a service or to access an object managed by a service, are assigned to roles. Separately, principals are associated with roles. The motivation is that users join, leave and change role in an organisation more often than the services that they use are changed.

At the Computer Laboratory we have developed OASIS, [3, 4] a parametrised RBAC system. OASIS provides a comprehensive rule-based means to check that users activate appropriate roles, thus acquiring the privileges that authorise them to use services. Collections of such rules are stored in OASIS policy files for role activation and service-specific authorisation. OASIS roles, and rules, are parametrised allowing fine-grained expression of policy. A role activation rule for a role r takes the form [4]

$$r_1, .., r_n, a_1, .., a_m, e_1, .., e_l \vdash r$$

where r_i are prerequisite roles, a_i are appointment certificates (most often persistent credentials) and e_i are environmental constraints. Environmental constraints allow restrictions to be imposed on when and where privileges can be exercised, for example at restricted times or from restricted computers. Any predicate for activating a role can be tagged as a *membership rule* and must remain true for the principal to remain active in the role. Such predicates must be monitored, and their violation immediately triggers revocation of the role and associated privileges from the principal. OASIS roles are activated within sessions. An authorisation rule for some privilege p takes the form

$$r, e_1, .., e_l \vdash p$$

OASIS has been used for multi-campus electronic courseware delivery [22], demonstration of a prototype Electronic Health Record service by Clinical and Biomedical Computing Limited (CBCL), and for general healthcare services in the IBHIS project [10]. The implementation of OASIS used by these projects uses a web service portal and is engineered as an EJB application.

Parametrised roles, together with environmental constraints, allow fine-grained policy requirements to be expressed and enforced [21]. For example *treating-doctor(doctor-ID, patient-ID,. . .)* makes the principals involved explicit when this is needed and allows the requirement that "the patient must be registered with the doctor", specified in the role activation policy, to be checked in a local registration database at role activation time. It also allows exceptions to be indicated, such as "my neighbour (who is a doctor) may not read my EHR". Current NIST and ANSI RBAC standards [1] do not support parametrisation.

4 The Middleware Communication Service

4.1 Evolution

Middleware aims to support distributed applications, i.e. interworking distributed components, and comprises a communications substrate and services. Traditional middleware, such as early RPC, CORBA and Java-RMI implicitly assumed a LAN infrastructure and embodied the synchronous, one-to-one, request-response invocation paradigm. Our Cambridge Event Architecture (CEA) of the early 1990s [2, 6] extended this object-oriented middleware style with *advertise, subscribe, publish-notify* communication. Later, CORBA and Java-based middleware offered a range of asynchronous extensions, yet none were sufficiently scalable. Traditional message oriented middleware, such as IBM's MQSeries, was asynchronous, one-to-one and reliable, supporting transactional database applications. With the advent of the internet-based worldwide web it became commercially essential to offer a web portal interface to applications. But the web services paradigm still embodies a client-server, many-to-one interaction style, in that a web service can handle many simultaneous clients, keeping their personal state private from each other. What is lacking in this evolution is a scalable multicast paradigm, part of the required toolset for building distributed applications. Although Jini was an attempt at supporting multicast, its tuple-space implementation proved hard to distribute for large-scale systems.

4.2 Publish/Subscribe

Traditional publish/subscribe systems, such as TIBCO's rendezvous, allow hierarchical "topics" to be advertised, subscribed to, published and notified. A large volume of data is transmitted efficiently over a wide area, via a network of brokers. The routing is based on the interests of subscribers and only the data subscribed to is delivered to the client. Publish/subscribe has removed the need for subscribers to know the names and addresses of publishers (via name servers). All that is needed is that topics of potential interest are advertised. This is potentially useful for building general distributed systems as well as its traditional applications: newsfeeds, stock-quotes and sports-reporting. Research projects have experimented with type/topic and content/attribute-based publish/subscribe systems, for example, Gryphon [11], Jedi [13], Elvin [29], Rebeca [16], Siena [12], Steam [20], and our local system, Hermes [23, 24, 26].

But multicast is not sufficient in itself: middleware communication should support the full range of interaction styles, yet direct, one-to-one communication between mutually known entities has been lost. Because of the inherent anonymity of publish/subscribe it is not possible even to reply to a publication, either anonymously (as in a request to vote) or identified.

We believe that integrating publish/subscribe with parametrised role-naming will provide a means of achieving scalable multicast and individual communication. For example, within a county police domain, many publications may be destined for named roles, for example messages to the role *officer-on-duty (. . .)* will reach the duty desks of all police stations. Multicast communication therefore seems a reasonable default. In certain cases however, a message may only be appropriate for some specified member of the role: *officer-on-duty (Cambridge-city, . . .)*, or even a named individual: *sergeant (ID, . . .)*.

We now outline how publish/subscribe is implemented for large-scale systems with a view to envisaging how the one-to-one and one-to-many paradigms might be integrated.

4.3 Publish/subscribe implementation

Large-scale, publish/subscribe messaging technology typically comprises a network of event brokers that provide a communication service, and lightweight clients that use the service to advertise, subscribe to and publish events. Such distributed systems are subject to failures of nodes and links, and their components may join and leave dynamically. A communication service must be robust under these conditions; fault-tolerant and dynamically reconfigurable. For this reason the event brokers are often built above a peer-to-peer overlay network, such as a distributed hash table (DHT), since peer-to-peer naming and protocols provide the necessary robustness. Publish/subscribe systems are classified as type/topic-based in which only the event name may be indicated in a subscription or, more expressively, content/attribute-based, which subsumes the former.

Our EDSAC21 research will be based on our Hermes publish/subscribe communication system [23, 24], which is used here as an example of a publish/subscribe implementation. Hermes is a distributed, content-based publish/subscribe architecture with an integrated programming model and strong event typing. It is built on a DHT-based peer-to-peer routing substrate to provide scalable event dissemination and fault tolerance.

A Hermes system consists of two kinds of component: *event brokers* and *event clients* (*publishers* and *subscribers*). Event brokers form the application-level overlay network that performs event propagation by means of a content-based routing algorithm. Event clients either publish or subscribe to events in the system. An event client has to maintain a connection to a *local event broker*, which then becomes *publisher-hosting*, *subscriber-hosting*, or both. An event broker without connected clients is called an *intermediate broker*.

Hermes supports *event typing*: every published event (or *publication*) in Hermes is an instance of an *event type*. Hermes event types are organised into inheritance hierarchies, so that some events at lower levels in a type hierarchy can satisfy some subscriptions at higher levels, depending on attribute values. An event type has an *event type owner*, an *event type name* and a list of typed *event attributes* so that, at runtime, publications and subscriptions can be type-checked by the system. Each event type defined within a domain is registered by its owner at a local event broker. Before a publisher can publish an event instance, it must submit an *advertisement* to its local event broker, containing the event type that it is willing to publish. Routing state can then be set up in the event brokers. Subscribers express their interest in the form of *subscriptions* that specify the desired event type and a conjunction of (content-based) filter expressions over the attributes of this event type.

To ensure that an event dissemination tree is created between publishers and subscribers, some broker in the system must become the *rendezvous node* for each particular event type. A rendezvous node is selected by hashing the event type name to a broker identifier — an operation that is supported by the peer-to-peer routing substrate [27]. Advertisements and subscriptions are routed towards the rendezvous node, and brokers along the path set up filtering state for them. For reliability reasons, there are several replicated brokers that act as rendezvous nodes for a given type. The rendezvous node keeps a copy of the event type definition that is cached at other brokers throughout the system for type-checking advertisements, subscriptions, and publications.

Most systems optimise content-based routing of events with a *subscription coverage relation*, stating which subscriptions are subsumed by others [12]. This allows brokers to reduce the number of events sent through the system by enabling them to filter non-matching events as close as possible to the publisher; these filters become increasingly specific as events approach subscribers.

If all brokers are trusted, messages can be encrypted on the wire and decrypted at brokers for routing purposes. If brokers are not uniformly fully trusted, so that some brokers are not trusted to see some attributes of some messages, then the optimised routing described above cannot always be carried out and a cruder approach must be taken.

4.4 Integration of Sensor Networks

Event-based communication is a natural paradigm for pervasive/ubiquitous computing. Many events may originate from badges, vehicles and sensors. Publish/subscribe is useful in that clients need not register interest with all possible event sources, and need only subscribe to a topic such as *bus-event (route-4, ...)*. In some cases, sensor data is at too low a level to be meaningful to applications and must be cleaned and aggregated before being passed to applications in the form of events. In the Cambridge TIME project (Transport Information Monitoring Environment) we aim to provide an event-based middleware layer on which applications can be built. Applications will be able to subscribe to, for example, bus events, taxi events, without having to know the details of the underlying sensor layer.

4.5 Event Patterns and Application Contexts

Isolated events may mean little to applications. What is needed may often be some higher level notion such as "a congested street" or "an empty meeting room" [19]. An interesting research area is to investigate the extent to which meaningful user requirements may be expressed as patterns of events, achieved through (bottom-up) event composition [25]. Higher level context models may instead need to be provided for clients, rather than a subscription and notification service for composite events. Mathematical specification of contexts and graphical representation to clients are being investigated.

4.6 Database Integration

As well as traditional database use, such as lookup and insertion, applications may wish to be informed when certain state changes occur. Examples are "tell me when any Cambridge garage has a Clio automatic in stock" or, for our police application, "tell me when a record relating to some particular suspect is recorded by any county police officer". We have already extended the triggers provided in the object-relational DBMS PostgreSQL to support fine-grain active predicates on the content of relations [8].

In the EDSAC21 project we are integrating these active predicates into the secure publish/subscribe framework. Event types are defined and managed exactly as for Hermes; attribute types are identified with SQL data types. Database administrators then advertise active database content conforming to these event types. For example, the Motor Traders Association can define an event type for specifying models of car. Any main dealer who wishes may then advertise *new-model* events conforming to this type, and subscribers will then be notified whenever new car models that interest them become available.

At database level, it is natural to implement such events through active (virtual) relations in some view of the database. Database administrators can then normalise the underlying physical database as they require. Security views are one of the traditional methods of implementing access control in relational databases, and associating roles with the access privileges to views is entirely consistent. In this approach, subscription to active database events has the same syntax and semantics as for other advertised events. Both event routing and secure access through RBAC can also be handled in the standard way, using Hermes and OASIS.

5 RBAC - Controlled Communication

The aim of the EDSAC21 project is to create a secure middleware suitable for a wide range of large-scale, widely distributed application domains. We aim to achieve this by using role-based access control, together with fine-grained data encryption with associated key management, integrated with extended publish/subscribe - based communication. The basic technologies, OASIS and Hermes, were designed for large scale, multi-domain, widely distributed systems, and it remains to extend Hermes and integrate it with active databases and OASIS.

RBAC can be used to secure any service; the service's authorisation policy indicates, for each method invocation, the role that must be proved to be active and any constraints that must be satisfied. RBAC may therefore be used to secure a communication service within a domain, controlling the roles that can advertise, publish and subscribe to each possible message type. Indeed, authorisation may be expressed at the level of individual message attributes. Inter-domain communication is achieved through negotiated agreements, expressed as access control policy, on which role(s) of one domain may receive (which attributes of) which messages of another.

If the communications substrate could be guaranteed secure and trustworthy, then RBAC would achieve precisely the visibility specified by policy. In practice, we have

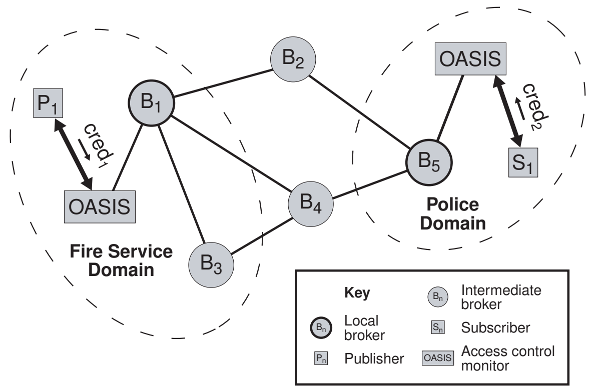

Figure 1. Overview of the EDSAC21 architecture

to protect confidential data on the wire and in the brokers by means of encryption.

In OASIS RBAC, within a naming and protection domain, the authorisation policy for any service specifies how each method can be invoked under access control, in terms of roles and environmental constraints. Here, we use OASIS to protect the publish/subscribe service in this way at a local event broker of the domain, as shown in Figure 1. The service's methods include *define(...), advertise(...), publish(...)* and *subscribe(...)*, where *define(...)* is used to register a message type with the service and specify its security requirements. This includes which attributes are confidential and must be encrypted. The *key policy* for a domain specifies which event brokers are allowed to access the attribute encryption keys associated with the messages. A secured publish/subscribe system also needs a *node access policy* that specifies the credentials necessary for a node to connect to the publish/subscribe system. Publishers, subscribers and brokers require public key pairs, bound to identity certificates (e.g. X.509 [18]), to connect to their local OASIS service.

As introduced in Section 4, the notion of role is ideally suited to a multicast communication style. For example, the London sales domain of a company may define a role *london-salesman* and might choose to define a message topic *for-london-salesmen (attribute-list)*. Subscription policy for the topic will indicate that all members of the role can subscribe to the topic, and which roles can publish it. We envisage that many other topics will be defined, for example *london-sales-report (...)*. Inter-domain communication is supported, after human negotiation, by indicating in policy that a specified role of some domain may subscribe to certain (attributes of) topics published by some other domain.

One-to-one communication can be seen as a special case, implemented using OASIS parametrised roles, e.g. a publication to be notified to an individual salesman would specify his ID: *london-salesman (salesID,...)*. In both cases, multicast or selection by parameter, a recipient may wish to respond to the message, and this is not supported in current publish/subscribe systems. We aim to generalise our design to support both anonymous and specified recipients of publications and the ability for a recipient to reply to the publisher either anonymously (as required in voting, for example) or identified.

We optimise routing through the overlay broker network for the one-to-one cases while retaining the scalability of publish/subscribe for large-scale multicast. If the communication between two entities is anonymous, we leverage multi-hop routing in the overlay network to decouple the sender from the receiver. This may be combined with cryptographic techniques, such as onion routing [17], to achieve strong anonymity guarantees. If anonymous communication is not required, a direct connection between the two local brokers hosting the communicating parties is created, thus bypassing the overlay network. This leads to an efficient implementation of one-to-one communication in our middleware, similar to the mechanism used in standard request/reply middleware. However, the overlay network gives the flexibility to return to multi-hop communication when the location (and thus the communication endpoint) of the receiver changes due to mobility.

If the communication substrate could be guaranteed 100% secure and trustworthy then RBAC and filtered message contents would achieve precisely the visibility specified by policy. In practice, we have to protect confidential data on the wire and in the brokers. Some brokers may be deemed more trustworthy than others. For example, the brokers within a police domain may be trusted by police to decrypt data for routing purposes. However, a compromised broker could inadvertently leak confidential publications and subscriptions to an attacker. Also, a single event might hold both public and confidential information. We therefore extend the security inside the publish/subscribe network by encrypting confidential event attributes to protect them from unauthorised access by brokers or clients. Key management is transparent to the clients of the publish/subscribe service because it is managed by their local brokers. This attribute-level encryption is a unique feature of our model.

6 Unstructured Global Computing

The EDSAC21 project aims to support domain-structured distributed computing. Mobile clients can be accommodated provided that they can connect securely to a broker in their home domain, where policy is stored and enforced, and can provide the required credentials or have them looked-up in a domain database. Mobile clients' messages can be buffered (managed by the clients' local brokers) while they are detached, and made available when they reconnect.

As small devices become increasingly prevalent, connected by MANETs (Mobile, Ad-hoc Networks), a less structured model of global computing emerges. Here, mobile users may be disconnected from their home domains, may encounter other users and services and may wish to interwork. There is no longer a substructure to support naming, authentication, access control etc. An entity may be named by some identifier or public key, generated at will. In this scenario the notion of trust becomes significant. Any entity may accumulate evidence about interactions with other named entities and may build up trust for them. On a request for interaction, an entity may request from others, with whom it is in contact, their accumulated experience of the potential partner. Threats come from the ease with which new identifiers can be generated and the potential for collusion between groups to build up high trust for each other.

If little is at risk through engaging in an interaction, trust need not be high. For a high-risk operation, a high level of trust may need to be established. The EU SECURE project has defined a trust model for this general scenario and has evaluated it for a number of applications such as electronic purse, grid-based file storage and spam email detection.

7 Conclusion

This paper has presented some approaches to tackling unsolved problems in distributed system design. Many other researchers are carrying out related work, but the focus here has been on various projects in Cambridge. As always in computing we are "chasing a moving target". New technology allows greater mobility of users and computers and ever-increasing scale.

Our knowledge of how to build large-scale, complex distributed systems is in its infancy and the tried and tested strategy of slow growth is not available to us. We do not have systems already in place that can evolve incrementally. Legislators may express policy that is beyond the scope of current systems to implement. Governments procure distributed systems, specified according to these requirements. It is necessary to determine which policy is essential, which is ill-advised and where logically conflicting requirements have been specified. It is also essential to ensure that systems are engineered in such a way that they can reflect changes in policy.

Open interoperability should be an essential part of most large-scale system specifications. No single company should be allowed a monopoly position. Ideally, it should be possible for expertise to be shared, real experts acknowledged and used, and strengths in different areas combined. We should be able to make it clear that some requirements cannot at present be met. Market forces and competition have led to disaster so many times that a new model of cooperation, or competition within a cooperative framework of interoperability, is surely needed for successful complex systems to be designed and deployed.

Acknowledgements

The Opera research group for all their ideas and work over the years. Support from the UK EPSRC under grants GR/S57303 ECCO: Event Brokering for a Distributed Adaptive Mobile Environment, and GR/T28164 EDSAC21: Event-driven, Secure Application Control for the 21st Century, also EU:IST-2001-32486 SECURE: Secure Interaction among Ubiquitous, Roaming Entities.

References

[1] American National Standards Institute, International Committee for Information Technology Standards (ANSI/INCITS), ANSI INCITS 359-2004, *see also http://csrc.nist.gov/rbac/*

[2] J. Bacon, J. Bates, R. Hayton, K. Moody, Using Events to build distributed applications, *IEEE Services in Distributed and Networked Environments (SDNE95)*, 1995, 148–155

[3] J. Bacon, K. Moody, W. Yao, Access Control and Trust in the Use of Widely Distributed Services, *Middleware 2001. Volume LNCS 2218., Springer-Verlag*, 2001, 300–315

[4] J. Bacon, K. Moody, W. Yao, A Model of OASIS Role-Based Access Control and its Support for Active Security, *ACM Transactions on Information and System Security (TISSEC), November*, 2002, 492–540

[5] J. Bacon, K. Moody, Towards secure, scalable middleware, *Comm. ACM, 46(5), June*, 2002, 59–63

[6] J. Bates, M. Spiteri, J. Bacon, D. Halls, Integrating Real-World and Computer-Supported Collaboration in the Presence of Mobility, *IEEE WET-ICE (Workshops on Emerging Technologies in Collaborative Environments) Workshop on Collaboration in the Presence of Mobility, Stanford, CA*, 1998, 256-261
Best paper award of workshop and conference

[7] A. Belokosztolszki, D.M. Eyers, P.R. Pietzuch, J. Bacon, K. Moody, Role-Based Access Control for Publish/Subscribe Middleware Architectures, *Proc. 2nd International Workshop on Distributed Event-Based Systems (DEBS'03) ACM SIGMOD, San Diego, USA*, 2003

[8] A. Belokosztolszki, D. Eyers, W. Wang, K. Moody, Policy Storage for Role-Based Access Control Systems. *IEEE WET-ICE (Workshops on Emerging Technologies in Collaborative Environments) Workshop on Enterprise Security, Linz, Austria*, 2003, 196–201

[9] A. Belokosztolszki, K. Moody. Meta-policies for distributed role-based access control systems. *Proc.3rd International Workshop on Policies for Distributed Systems and Networks (Policy 2002), Monterey, CA, USA*, June 2002, 106–115

[10] K. Bennett et al., A Broker Architecture for Integrating Data Using a Web Services Environment, *Proc. 1st International Conference on Service-Oriented Computing, Trento, Italy*, Dec 2003

[11] S. Bhola, R.E. Strom, S. Bagchi, Y. Zhao, J.S. Auerbach, Exactly-Once delivery in a Content-Based Publish/Subscribe System, *Proc. Dependable Systems and Networks (DSN), Bethesda MD, USA*, 2002, 7–16

[12] A. Carzaniga, D.S. Rosenblum, A.L. Wolf, Design and Evaluation of a Wide-Area Event Notification Service, *ACM Trans. on Computer Systems, 19*, 2001, 332–383

[13] G. Cugola, E.D. Nitto, A. Fuggetta, The JEDI Event-Based Infrastructure and its Application to the Development of the OPSS WFMS, *IEEE Transactions on Software Engineering (TSE) vol.27*, 2001, 827–850

[14] P.T. Eugster, P.A. Felber, R. Guerraoui, A.M. Kermarrec, The Many Faces of Publish/Subscribe, *ACM Computing Surveys 35*, 2003, 114–131

[15] D.F. Ferraiolo, D.R. Kuhn, Role Based Access Control, *15th National Computer Security Conference*, 1992

[16] L. Fiege, M. Mezini, G. Muehl, A.P. Buchmann, Engineering Event-based Systems with Scopes, *Proc. 16th European Conference on Object-Oriented programming (ECOOP), Springer Verlag*, 2002, 309–333

[17] D. Goldschlag, M. Reed, P. Syverson, Onion Routing for Anonymous and Private Internet Connections, *Comm. ACM, 42(2), February*, 1999, 39–41

[18] ITU-T International Telecommunication Union: ITU-T recommendation X.509, 2000

[19] E. Katsiri, J. Bacon, A. Mycroft, An Extended Publish/Subscribe Protocol for Transparent Subscriptions to Distributed Abstract State in Sensor Driven Systems using Abstract Events, *Proceedings International Workshop on Distributed Event-Based Systems, May 24-25, Edinburg,UK*, 2004, 68–73

[20] R. Meier, V. Cahill, STEAM: Event-based Middleware for Wireless Ad Hoc Networks, *Proceedings, 1st International Workshop on Distributed Event-Based Systems (DEBS'02), Vienna, Austria, IEEE*, 2002, 639–644

[21] K. Moody. Coordinating policy for federated applications. *In 14th IFIP WG3 annual working conference on Data and Application Security, Schoorl, Netherlands 2000. Proceedings Kluwer* 2001, 127–134

[22] S. Neely, H. Lowe, D. Eyers, J. Bacon, X. Gong, An architecture for supporting vicarious learning in a distributed environment, *Proc. ACM Symposium on Applied Computing*, 2004, 963–970

[23] P.R. Pietzuch, J. Bacon, Hermes: A Distributed Event-Based Middleware Architecture, *Proceedings, 1st International Workshop on Distributed Event-Based Systems (DEBS'02), Vienna, Austria, IEEE*, 2002, 611–618

[24] P.R. Pietzuch, J. Bacon, Peer-to-Peer Overlay Broker Networks in an Event-Based Middleware, *Proceedings, 2nd International Workshop on Distributed Event-Based Systems (DEBS'03). ACM SIGMOD, San Diego, USA*, 2003

[25] Peter R. Pietzuch, Brian Shand, Jean Bacon, A Framework for Event Composition in Distributed Systems. *In M. Endler and D. Schmidt, editors, Proceedings, 4th ACM/IFIP/USENIX Int. Conf. on Middleware (Middleware '03), Rio de Janeiro, Brazil, Springer*, 2003, 62-82
Best paper award

[26] P.R. Pietzuch, Hermes: A scalable event-based middleware *University of Cambridge PhD thesis and TR590*, 2004

[27] A. Rowstron, P. Druschel, Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems, *Proceedings Middleware 2001, Volume LNCS 2218, Springer*, 2001, 329–350

[28] R. Sandhu, E. Coyne, H.L. Feinstein, C.E. Youman, Role-based access control models, *IEEE Computer, 29(2), February*, 1996, 38–47

[29] B. Segall, D. Arnold, J. Boot, M. Henderson, T. Phelps, Content-Based Routing with Elvin4, *Proc. AUUG2K, Canberra, Australia*, 2000