Dynamic Access Control in a Content-based Publish/Subscribe System with Delivery Guarantees

Yuanyuan Zhao
IBM T. J. Watson Research Center
yuanyuan@us.ibm.com

Daniel C. Sturman
IBM Software Group
sturman@us.ibm.com

Abstract

Content-based publish/subscribe (pub/sub) is a promising paradigm for building asynchronous distributed applications. In many application scenarios, these systems are required to provide stringent service guarantees such as reliable delivery, high performance, high availability and dynamic system security.

In this paper, we address the issue of dynamic access control in a content-based system that provides reliable delivery and high availability through redundant routes. We define a deterministic service model of dynamic access controls that enables precise control over event confidentiality. Under this model, the semantics of reliable delivery is clearly defined, that is, the messages delivered in response to the same subscriptions from pub/sub clients running on behalf of the same principal will be exactly the same, regardless of their connecting locations, network latency and failures. We present an algorithm that implements this service model. The algorithm is efficient and highly available in that it enables uniform enforcement of access control and enables content-based routing to choose any path from among several redundant routes without requiring consensus among the brokers.

1. Introduction

Content-based publish/subscribe (pub/sub) messaging is a popular paradigm for building asynchronous distributed applications. A content-based pub/sub system consists of *publishers* that generate messages and *subscribers* that register interest in messages matching the *predicate/Boolean filter* specified in their subscription. The system ensures timely delivery of published messages to all interested subscribers, and typically contains *routing brokers* for this purpose. Publishers and subscribers are both clients of the system and are decoupled from each other ([12]).

For many applications, content-based pub/sub systems are required to provide strong service guarantees (such as reliable, in-order, gapless delivery [7, 8]), high scalability to support a large number of clients, high service availability and high performance/throughput. In order to achieve these goals, typical systems 1) propagate and consolidate subscription information toward publishers; 2) using the subscription information, perform content filtering to achieve good network bandwidth utilization and scalability; 3) and utilize redundant network paths for high service availability. Related works in this area include [11, 20, 5, 27, 24, 26, 10, 23].

The issue of access control in pub/sub systems has not received much attention, probably due to the fact that security *seems* contrary to the pub/sub philosophy of anonymity and decoupling of information producers and consumers [17]. A further challenge comes from the requirement to allow access control policies to change dynamically without disrupting the pub/sub service. Systems should be able to continue functioning without having to shut down to enact new access control policies. Such a requirement is often demonstrated by mission critical applications such as electronic trading in financial markets. There is as yet no published definition of exactly when access control changes are specified to take place and how the changes should affect content delivery, especially deliveries with reliability guarantees and high availability using redundant routing paths.

In this paper, we address the issue of providing dynamic access control in a pub/sub system with content-based filtering and routing, reliable delivery and redundant routes. Our contributions are:

- A deterministic service model of dynamic access control in content-based pub/sub systems. The deterministic guarantee enables precise control over event confidentiality and is independent of issues like client location, network latency and failures.
- A novel algorithm supporting this deterministic service model. Using this algorithm, access control changes are performed uniformly across all brokers to which the affected principals connect. There is no need for the system to obtain consensus from these brokers,

which could compromise the efficiency of the system and timeliness of enacting the change.

- Performance evaluation of the algorithm in the context of an industrial strength pub/sub system - Gryphon.

The rest of the paper is organized as follows: Section 2 defines the deterministic service model. Section 3 describes the environment in which our guarantee will be implemented, namely a redundant routing network, subscription propagation, content-based routing and reliable delivery. Section 4 describes an algorithm that implements this service model. Section 5 describes implementation and experimental results. Section 6 is an overview of related works and we conclude in Section 7.

2 Deterministic Service Model

We present in this section a deterministic service model of dynamic access control. We describe the various entities involved in dynamic access control and their roles, a content-based form for specifying access control rules and the clear starting points of access control changes.

2.1 System Entities & Content-based Rules

In our service model, there are two types of entity that are involved in access control.

Security administrator The security administrator is the ultimate authority of access control in the system. The security administrator decides (based on external factors such as client service contracts) the access rights for client principals (defined below) and/or whether there should be any change to their existing access rights. The security administrator instructs the system of his/her decisions through an administrative interface.

In a large system, there may be multiple security administrators. As the changes made by each administrator may affect overlapping sets of clients, the system should accept the changes in a transactional and serializable manner. For the purpose of this paper and simplicity of discussion, we consider the security administrators as an abstract single entity that initiates a single sequence of policy changes.

Client principals Clients in our system have associated principals which are decided/verified by the system through authentication when clients connect. A client can connect to the system, publish messages or subscribe and receive messages. The access rights can also include the ability to advertise publications. Due to space constraint, we do not discuss it here. The client's capability to connect, publish and/or subscribe/receive messages is regulated by the access rights of its principal. The access control rules in our system are associated with principals. Multiple pub/sub clients running on behalf of the same principal can connect at different places in the system.

There are two types of principal in our system, *group* and *individual*. A group principal is a collection of individuals or recursively, other group principals. Access rights granted to a group principal are automatically granted to all members of the group, and recursively to the members of a member group.

Content-based form of Access Rights The access rights of a principal include the right to connect, the right to publish and the right to subscribe to and receive messages. We adopt a content-based form for specifying access control rules of these three rights. An access control rule takes the following form of a tuple of three elements:

[Principal, Access type, Content filter]

A rule of such form specifies that a principal has the right to connect to the system, publish or subscribe to messages matching a content filter. While publish and subscribe rules can take a non-trivial filter, connect rules are specified with $true$ or $false$ to indicate the right to connect or not. For example, the rules that allow a principal *John Doe* to connect and subscribe to stock quotes are specified as follows:

[John Doe, Connect, True]
[John Doe, Subscribe, type='quote']

A pub/sub client on behalf of a principal is allowed to publish messages that match the publish rules of its principal and is allowed to receive messages that match BOTH its subscription filters and the subscribing rules of its principal. This allows the system to provide 1) **information authenticity** by allowing only authorized sources to publish messages; 2) **information confidentiality** by only distributing messages to authorized subscribers; 3) **protection against denial-of-service (DoS) attacks** initiated by malicious subscribers who request a large number of messages that are only going to be discarded. This large number of messages can result in congestion in the network and impair the system's capability to serve other clients.

Group Access Control Group and individual principals share the same form of connect, publish and subscribe access rights. In addition, a new type of rule, *member list*, exists for group principals. For example, a *premium subscribers* group that includes *Jane Smith* and *James Brown* and has subscribing rights to all stock quotes, news and reports has the following access control rules:

[Premium group, Member list, {Jane Smith, James Brown}]
[Premium group, Subscribe, type='quote' or type='news' or type='report']

All members in a group are automatically granted the access rights of the group. Thus, the access rights of an individual principal are the union of the individual's rights and the rights of all groups it belongs to. Hence, *Jane Smith* and *James Brown* will have access to all stock quotes, news and reports in addition to other access rights they are granted.

2.2 Clear Starting Points of Access Control Changes

We present in this section our deterministic service model that provides clear starting points for access control changes. In this model, access control rules/changes are initiated by the security administrator at the administrative console and stored into a persistent storage called **ACL DB**. At any time, the security administrator may specify a number of changes pertaining to one or more principals. These changes are considered as a batch that must be enforced atomically. After the security administrator confirms each batch of changes, the changes are propagated throughout the broker network.

A broker can host one or more message streams. The publishers can connect to any broker in the system and are assigned to any stream by the broker. Each stream contains in-order the messages published by one or more publishers. For each of these streams, the broker picks a starting point to enact the new access control rules. The starting point is chosen in a way such that: 1) successive batches of changes get later starting points; and 2) the starting point is late enough so that no messages after the starting point could have been delivered according to the old rule. This constraint can be easily achieved by designating a newly published message on the stream as the starting point. The starting point information is sent back to the security administrator for future inquiries and references. The new rules are enforced uniformly throughout the system on all messages after the starting points, no matter where the pub/sub clients on behalf of the affected principal(s) connect.

We illustrate the effect of an access control policy change using an example in which a principal *John Doe*'s subscribing rights went through 3 phases of changes: 1. *John Doe* became a member of the promotional group which had subscribing access only to stock quotes; 2. *John Doe* became a premium subscriber and subsequently gained subscribing access to all three types of financial information; 3. *John Doe*'s premium subscription expired and as a result he lost subscribing rights to financial news and reports.

Shown in Figure 1, a subscriber on behalf of principal *John Doe* connected to the system and requested a subscription of issue='ibm'. Under the service model, every time the access rights of *John Doe* changes, the system provides a clear starting point in each message stream such that 1) a message *before* the starting point is delivered to the client if and only if the message satisfies both the subscription filter and access right filter *before* the change; and 2) a message *after* the starting point is delivered to the client if and only if it satisfies both the subscription filter and access right filter *after* the change. In the stream in our example, if the starting points chosen are message 100 for the first access change, message 103 for the second access change and message 106 for the third change, the messages delivered to the client will be $100, 103, 104, 107, 109$. Notice that non-quotes are only delivered in the range $[103, 105]$. In a system that has more than one message stream, this activity happens to all streams, each with its individual start points.

3 Environment

In this section, we describe 1) a topology model of redundant routing networks that are deployed for high availability; 2) reliable delivery; and 3) the techniques such as subscription propagation and content-based routing that are usually employed for performance and scalability. These techniques are also used as building blocks to simplify the description of the algorithm.

3.1 Routing Topology

We adopt an *abstract* topology model of spanning trees of *nodes* where each node includes multiple *virtual brokers* that are redundant and can work interchangeably. Trees are noncyclic structures that simplify the task of loop-free routing. Tree nodes with redundant brokers provide high availability.

We refer to a broker where publishers connect as a publisher hosting broker (PHB) and a broker where subscribers connect as a subscriber hosting broker (SHB). For simplicity, we discuss our work from the standpoint of one PHB. The abstract network may be constructed such that any physical broker hosting clients implements a *virtual broker* in a leaf node. As a result, the SHBs only reside in the leaf nodes; and there is only one PHB and it resides in the root node. The direction *upstream/downstream* points toward/away from the root. As a client connects to one broker, each leaf node contains one broker.

This topology model can represent a large range of practical topologies as one can transform a graph with redundant paths into a topology under this model by grouping brokers into tree nodes and inter-broker links into tree edges. Figure 2 provides an example. Details on how to form a virtual broker network from a physical broker network is out of the scope of this work and can be found in [7].

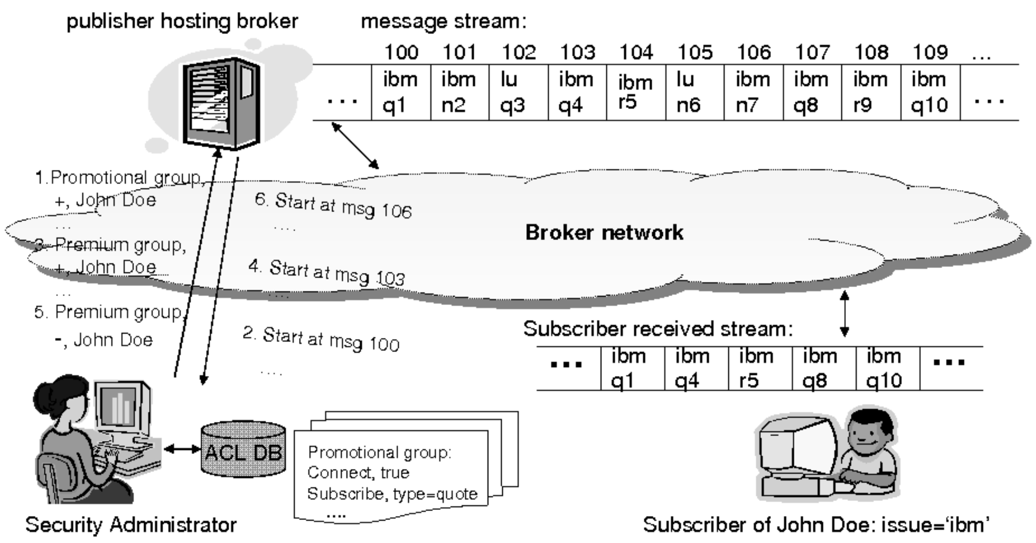
publisher hosting broker
message stream:
100 101 102 103 104 105 106 107 108 109
ibm q1 | ibm n2 | lu q3 | ibm q4 | ibm r5 | lu n6 | ibm n7 | ibm q8 | ibm r9 | ibm q10
1.Promotional group, +, John Doe
3. Premium group, +, John Doe
5. Premium group, -, John Doe
6. Start at msg 106
4. Start at msg 103
2. Start at msg 100
Broker network
Subscriber received stream:
ibm q1 | ibm q4 | ibm r5 | ibm q8 | ibm q10
ACL DB
Promotional group:
Connect, true
Subscribe, type=quote
Security Administrator
Subscriber of John Doe: issue='ibm'

Figure 1. Service Model of Dynamic Access Control:'q' for quotes, 'n' for news & 'r' for reports.

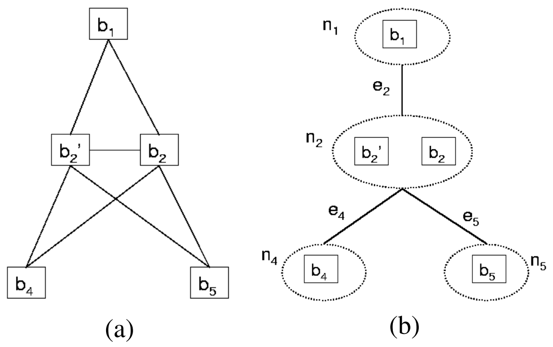

Figure 2. Redundant Routing Networks

3.2 Subscription Propagation, Content-based Routing & Reliable Delivery

A valid implementation of access control can be one in which the PHB and intermediate brokers forward all published messages that match client subscriptions to SHBs, and SHBs enforce access control by delivering messages that match not only a client's subscription but also its access rights. Such a solution will be a perfectly correct implementation, but it may waste considerable bandwidth sending messages that will be later discarded.

Subscription propagation is an optimization which may result in fewer wasted messages being sent to SHBs in exchange for requiring the PHB and intermediate brokers to acquire knowledge about subscription predicates and perform filtering. By propagating clients' access rights along with their subscriptions, further savings in communication cost may be achieved.

Providing the deterministic service guarantee described in Section 2 is challenging in a content-based system deployed over a network with redundant paths. Due to content-based routing, gaps can not be detected by traditional methods such as publisher-assigned sequence numbers because each subscriber may request a completely unique sequence of messages to be delivered. Reliability in a content-based system hence requires brokers on the routing path to assist in gap detection ([7]).

In [29] and [28], we described protocols for subscription propagation. These protocols preserve reliable delivery and enable free routing choices on any of the redundant paths for system availability and load sharing.

In the next section, we use the reliable delivery and subscription propagation protocols as building blocks for constructing an efficient and highly available distributed protocol that enforces the deterministic semantics of dynamic access control to pub/sub clients. We adopt a domain-based trust model. All brokers within the same domains trust each other. Brokers that do not trust each other should be put into different domains and cross-domain communication is regulated by assigning access control rules according to their trust levels. For simplicity, we discuss the protocols under one trusted domain. This is of practical use as in a lot of commercial cases, pub/sub systems are deployed in a managed environment under the complete control of an administrator. The work can be extended to multiple trusted domains by treating a domain as a special pub/sub client and assigning a principal to the domain. The clients connected to the system through an un-trusted domain can only access messages that satisfy both the domain's right and their own access right.

4 Protocol

Our protocol provides the deterministic service guarantee of message delivery through 1) distributing access control information to brokers that host relevant principals; 2) restricting publishing activities by accepting only messages satisfying the publisher's publishing rights; 3) restricting client subscriptions using their subscribing rights; 4) propagating restricted subscriptions and hence enforcing access control in the routing brokers by performing content filtering on both the clients' subscriptions and access rights; 5) final enforcement of access control at the SHB. We describe each of these aspects.

4.1 Client Access Control Information

As previously mentioned, access control policies are maintained in a persistent storage called *ACL DB*. The security administrator makes policy changes in transactional batches to the ACL DB. Access control policies are associated with a control version, which is an integer counter. Each transactional batch brings the ACL DB into a new control version. The new access control rules are assigned with the new version number. As old access control rules may still be in effect for some messages, the ACL DB contains a mixture of access control rules with different versions. To avoid sending the whole state, the ACL DB distributes the new version of access control by publishing it as an incremental change.

Each PHB/SHB maintains a cache of latest access control rules for clients that are currently connected. When a client on behalf of a new principal connects, the broker retrieves an initial version of access control rules for the principal through a request/reply protocol with the ACL DB. The broker also establishes a subscription for receiving future access control changes for the principal. We use the subscription propagation and reliable delivery service in Gryphon to ensure the broker receives every access control change after obtaining an initial version of access rules for a connected principal.

When a PHB receives a new version of access control rules, it updates its cache. The PHB picks a starting point for the new version as the next message that will be published. A newly published message will only be accepted if it matches the publishing rights in effect. In addition, newly published messages are transmitted in the system carrying the access control version that is in effect.

We treat access control information as another type of information that can affect message routing in addition to client subscriptions. Thus, instead of propagating the original client subscription filters to the rest of the network, our SHBs propagate a restricted form of filters that are the intersection of the client subscription filter and the latest version of content-based access rules in the SHB's cache. When the access control rules change, the SHB re-computes the restricted subscriptions for all affected clients/principals with the new version of rules. The resulting subscriptions are propagated upstream atomically together with the control version. The upstream routing brokers handle the subscriptions without having to know whether the subscriptions are restricted. The upstream routing broker only needs to maintain a vector of control versions for each SHB in its downstream.

4.2 Routing Data Messages

As we propagate restricted filters, content-based routing is based on the intersection of client subscription filters and access control rules. This allows the routing brokers to participate in access control as well as the SHBs.

As mentioned in Section 4.1, a message in the system carries the control version that is in effect for the message. When routing the message for a downstream, a routing broker compares this version of the message with the sub-portion of its control version vector for SHBs located in the downstream route. The message is only filtered out if it does not match the restricted subscriptions from the downstream route and every element of the sub-portion of the broker's control version vector is no less than that of the message. In the case that the broker does not have a sufficiently large control version vector, it may conservatively send the message on that downstream route.

4.3 Enforcing Access Control at SHBs

The ultimate enforcers of access control are the SHBs as intermediate routing brokers may conservatively send messages that do not match a subscriber's access rights.

The SHB first examines whether it has received the access control rules of the version required by the message. If not, the SHB delays the processing of the message until the version of the access control rules arrives. If the SHB has received the control version of the rules, the SHB examines each restricted subscription that matches the message. If the restricted subscription has the same control version as the message, it is delivered to the subscriber. Otherwise, the message is not delivered to the subscriber.

4.4 Discussion of Efficiency

The use of control versions not only allows our algorithm to implement the clear starting point feature of our model, but also allows the system to be more asynchronous and fault tolerant. The distribution of access control changes with a control version number allows each broker in the system to proceed asynchronously instead of waiting for a slow or crashed broker if a transactional session of broadcasting to all brokers is utilized. Even in the case that a majority of brokers fail in a routing tree node, new access control rules can be enacted and the remaining broker can participate in enforcing access control without having to obtain an agreement from its redundant peers. When a broker recovers, even when its control version may lag behind, the broker can still participate in message routing, utilizing its part of the network capacity that would otherwise stay idle.

The use of a control rule cache of only connected principals allows the system to scale even in the large scale environment where the number of principals is large. The SHBs only needs to know access control rules for the principals whose clients are locally connected.

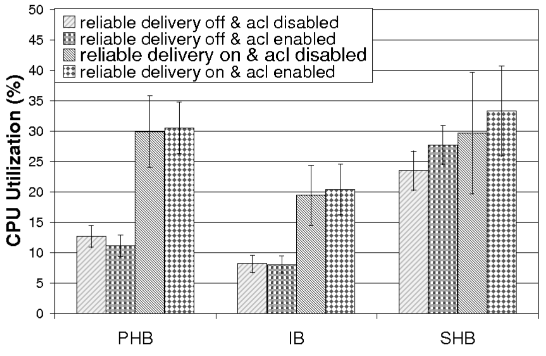

Figure 3. System Load in Steady State

5 Experimental Results

We have implemented our algorithm in the Gryphon system. We present some of the experimental results such as the cpu overheads and various latency metrics. We focus on the metrics related to subscribing access control.

5.1 Test Environment

Our testbed is a set of RS6000 F80 machines with six 500MHz processors and 3G RAM connected through a gigabit switch. The broker is implemented in Java with native I/O library and running in IBM JRE 1.4.

We used two versions of Gryphon for our experiments: one implements the protocol in this paper and another without any dynamic access control mechanism.

5.2 Experiment Results

System Load in Steady State This test compares the system performance of enforcing access control for subscribers when their subscribing rights do not change. We use a setup in which a publisher hosting broker pb is connected to an intermediate broker ib, which in turn connects to a subscriber hosting broker sb. The steady access control policy in this test permits the subscribers to receive all messages they subscribe to. Therefore, the broker network routes and delivers the same amount of messages as in the system when there is no support for dynamic access control. We examine the CPU overhead at each broker and compare it with our baseline results, i.e., results obtained in the Gryphon system with no dynamic access control support. This comparison represents the overhead incurred at brokers playing different roles in the protocol - access control version setter (pb), in-network access control enforcer(pb&ib) and end-point ultimate access control enforcer (sb).

Messages are injected into the system through pb at a rate of 2000 messages per second. We evaluate both the cases when the Gryphon special reliable delivery protocol is turned on or bypassed. To eliminate the impact of different file systems used for PHB persistent message streams, we perform Gryphon message logging but do not sync to the disk. The output message rate at the SHB is 20000 messages per second to 10000 subscribers. Figure 3 shows the CPU utilization at each of the brokers. This test shows that the CPU overhead increase at pb and ib are very small as the overhead is mostly for assigning and/or comparing control versions. The CPU increase at sb is higher due to the final access control enforcement that is performed against every matching subscriber for every message.

Latency Measurements We examine four latency metrics: 1) The latency of message delivery during steady state when there are no access control changes. 2) The latency of starting delivering messages to a newly connected subscriber when there are already connected subscribers at the SHB on behalf of the same principal p_1 and the new subscriber uses the *same subscription* as an existing subscriber. We call this the *local* delivery start latency. 3) The latency of starting delivering messages to a newly connected subscriber on behalf of a new principal p_2, however the new subscriber uses the same subscription as some of the existing subscribers at its SHB. As a result, the SHB does not have access control rules for the new subscriber cached and must retrieve initial access control rules. However, the SHB can process the new subscription *locally* without having to propagate the subscription outside to other brokers. We refer to this metric as the *new principal local* delivery start latency. 4) The latency of starting delivering messages for a newly connected subscriber on behalf of a new principal p_3 AND the new subscriber uses a new subscription that is not used by any other subscribers at the SHB. We call this the *new principal remote* delivery start latency as the SHB needs to propagate the subscription remotely to the PHB in addition to retrieving the initial access control rules.

We use a linear topology consisting of a PHB pb and a

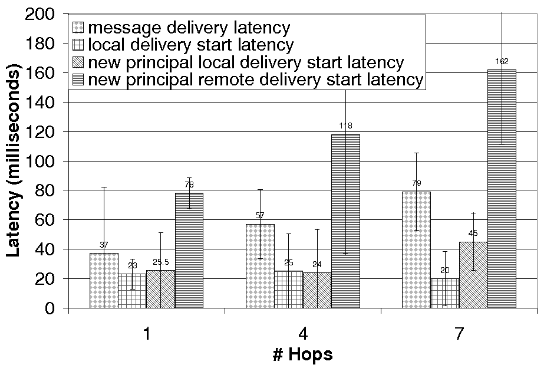

Figure 4. Latency Metrics

SHB sb that are connected through one or more hops of intermediate brokers $ib_{1..n}$. We colocate the ACL DB with pb. Thus, in order to receive initial access control rules for a new principal, sb has to communicate a round trip to where pb resides.

We measure message delivery latency using a latency sampler that publishes messages through pb and subscribes to its own messages at sb. Delivery start latencies are measured as the time between a subscriber submits its subscription and when it receives the first message.

Figure 4 shows the latency results with their standard deviations shown in error bars when there are 1, 4 and 7 hops from pb to sb. In this test, the message delivery latency and the remote delivery start latency for new principals increase linearly as the hop count increases. The local delivery start latency does not increase with the hop counts. The new principal local delivery start latency does not increase until 7 hops, because the initial acl retrieval is done at subscriber connection time and thus runs in parallel with the subscriber submitting its subscription. In the case of 1 and 4 hops, the initial acl retrieval is able to complete before the subscriber submits its subscription.

6 Related Work

A great volume of work on security issues in distributed messaging systems is in secure group communication systems ([14, 15]). In these systems, access control is usually provided by using a shared key among group members. To deal with a group member joining and leaving and protect information from the leaving members, keys must be changed. The focus of the works in this area is on group key management ([19, 9, 22]).

As pointed out by Opyrchal et al. [17], the dynamic nature of a content-based system makes the secure group communication approach infeasible for enforcing access control in a content-based system. The number of potential groups for n subscribers is 2^n and managing keys for these groups is expensive. What's more, the matching groups can change for each event, constantly changing encryption keys significantly slows down the throughput of common encryption algorithms such as DES [18]. Opyrchal et al. tackle the problem using group clustering and key caching.

Wang et al. [25] analyze security issues and requirements in Internet-scale pub/sub systems and presents directions to possible solutions to the various problems. They presented novel security problems of information and subscription confidentiality in an un-trusted pub/sub system and pointed out that methods on *computing with encrypted data* [2] and *secure circuit evaluation* [1] can be adapted to solve these problems. In their work, there is no discussion on how access to particular events can be controlled and enforced.

Belokosztolszki [6] presented a role-based model for access control [13] in content-based pub/sub systems. They integrate the OASIS [4] role-based access control system into the Hermes pub/sub middleware framework and point out that access control can be enforced as restrictions on the subscription filters. By leveraging the existing pub/sub platform, access control rules can dynamically change and be distributed to brokers that host clients. Bacon [3] extends the work to multiple trusted domains.

Miklos [16] devotes significant attention to specifying maximum and minimum security restrictions by ways of covering relations between filters, advertisement and events. The intuition is to use maximum security to restrict clients from accessing events they are not authorized and use minimum security to limit the overhead of doing too much content matching against too specific subscriptions.

Srivatsa and Liu [21] propose using keys, signatures and security guards to provide information confidentiality, integrity and authenticity and to fend off DoS attacks.

Existing work on access control in distributed messaging systems has focused on secure distribution of events. There has not been work on the semantics of dynamic access control with regard to reliable delivery. Our work in this area addresses this issue. Existing work on the secure distribution of events according to their access control rules is complementary to our work.

7 Conclusions

In this paper, we defined a deterministic service model for dynamic access control in content-based pub/sub systems. We presented a novel algorithm that implements this service guarantee while preserving the correctness of reliable delivery. The algorithm is efficient and highly available in that it pushes the enforcement of access control close to the event sources and enables content-based routing to utilize any path in a redundant routing network without requiring consensus among the brokers.

Acknowledgements The authors thank Dr. Sumeer K. Bhola for the insightful discussions on some part of the work and Dr. Rob E. Strom for reviewing parts of the initial draft. We also thank Prof. Jean Bacon and Lauri Pesonen for their insightful comments and generous help in shepherding the camera-ready version. The anonymous reviewers have provided some of the most useful comments.

References

[1] M. Abadi and J. Feigenbaum. Secure circuit evaluation. *Journal of Cryptology*, 2(1):1–12, 1990.

[2] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. *Journal of Computer & System Sciences*, 39(1):21–50, 1989.

[3] J. Bacon, D. M. Eyers, K. Moody, and L. I. W. Pesonen. Securing publish/subscribe for multi-domain systems. In G. Alonso, editor, *Middleware*, volume 3790 of *Lecture Notes in Computer Science*, pages 1–20, Grenoble, France, Nov. 2005. Springer.

[4] J. Bacon, K. Moody, and W. Yao. A model of Oasis role-based access control and its support for active security. *ACM Transactions on Information and System Security*, 5(4):492–540, 2002.

[5] S. Baehni, C. Chabra, and R. Guerraoui. Frugal event dissemination in a mobile environment. In *Proceedings of the ACM/IFIP/USENIX 6th International Middleware Conference*, 2005.

[6] A. Belokosztolszki, D. M. Eyers, P. Pietzuch, J. Bacon, and K. Moody. Role-based access control for publish/subscribe middleware architectures. In *Proceedings of the International Workshop on Distributed Event-Based Systems*, 2003.

[7] S. Bhola, R. Strom, S. Bagchi, Y. Zhao, and J. Auerbach. Exactly-once delivery in a content-based publish-subscribe system. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'2002)*, 2002.

[8] S. Bhola, Y. Zhao, and J. Auerbach. Scalably supporting durable subscriptions in a publish/subscribe system. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'2003)*, 2003.

[9] J.-C. Birget, X. Zou, G. Noubir, and B. Ramamurthy. Hierarchy-based access control in distributed environments. In *Proceedings of the IEEE International Conference on Communication*, 2001.

[10] A. P. Buchmann, C. Bornhövd, M. Cilia, L. Fiege, F. C. Gärtner, C. Liebig, M. Meixner, and G. Mühl. Dream: Distributed reliable event-based application management. In *Web Dynamics*. 2004.

[11] A. Carzaniga, D. Rosenblum, and A. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

[12] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[13] D. Ferraiolo and R. Kuhn. Role-based access controls. In *15th NIST-NCSC National Computer Security Conference*, 1992.

[14] P. Judge and M. Ammar. Security issues and solutions in multicast content distribution: A survey. *IEEE Network Magazine*, Januaray/February 2003.

[15] P. S. Kruus. A survey of multicast security issues and architectures. In *Proceedings of the 21st National Information Systems Security Conference*, 1998.

[16] Z. Miklos. Towards an access control mechanism for wide-area publish/subscribe systems. In *Proceedings of International Workshop on Distributed Event-Based Systems*, 2002.

[17] L. Opyrchal and A. Prakash. Secure distribution of events in content-based publish subscribe systems. In *Proceedings of the 10th USENIX Security Symposium*, 2001.

[18] F. I. P. S. Publication. Data encryption standard, 1977.

[19] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Computing Surveys*, 35(3):309–329, 2003.

[20] A. Rowstron, A. Kermarrec, M. Castro, and P. Druschel. Scribe: The design of a large-scale event notification infrastructure. In *Proceedings of 3rd International Workshop on Networked Group Communication (NGC 2001), UCL, London, UK*, November 2001.

[21] M. Srivatsa and L. Liu. Securing publish-subscribe overlay services with eventguard. In *Proceedings of the 12th ACM Conference on Computer and Communication Security*, 2005.

[22] Y. Sun and K. J. R. Liu. Scalable hierarchical access control in secure group communications. In *Proceedings of the 23rd Conference of the IEEE Communications Society*, 2004.

[23] D. Tam, R. Azimi, and H.-A. Jacobsen. Building content-based publish/subscribe systems with distributed hash tables. In *Proceedings of 1st International Workshop on Databases, Information Systems and Peer-to-Peer Computing*, 2003.

[24] A. Virgillito. *Publish/Subscribe Communication Systems: from Models to Applications*. PhD thesis, Universita degli Studi di Roma "La Sapienza", November 2003.

[25] C. Wang, A. Carzaniga, D. Evans, and A. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9*, 2002.

[26] Y.-M. Wang, L. Qiu, C. Verbowski, D. Achlioptas, G. Das, and P. Larson. Summary-based routing for content-based event distribution networks. *SIGCOMM Computer Communications Review*, 34(5):59–74, 2004.

[27] E. Yoneki and J. Bacon. An adaptive approach to content-based subscription in mobile ad hoc networks. In *Proceedings of 2nd IEEE Annual Conference on Pervasive Computing and Communications, Workshop on Mobile Peer-to-Peer Computing*, 2004.

[28] Y. Zhao, S. Bhola, and D. Sturman. A general algorithmic model for subscription propagation and content-based routing with delivery guarantees. Technical report, RC23669, IBM Research, 2005.

[29] Y. Zhao, D. Sturman, and S. Bhola. Subscription propagation in highly-available publish/subscribe middleware. In *ACM/IFIP/USENIX 5th International Middleware Conference (Middleware 2004)*, 2004.