

Flexible Secure Multicasting in Active Networks*

Zhaoyu Liu, R. H. Campbell, S. K. Varadarajan, Prasad Naldurg, Seung Yi, M. D. Mickunas
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{zhaoyu, roy, svaradar, naldurg, seungyi, mickunas}@cs.uiuc.edu

Abstract

In this paper we describe an alternative, flexible approach to multicast security in active networks. Traditional schemes for securing multicast communication have key management and scalability problems for many typical applications. In addition, traditional mechanisms are not capable of expressing flexible, situational security policies for multicast sessions and participants. Our scheme exploits the computational power of active networks to provide dynamic, flexible security for multicast applications. One of the main advantages of our scheme is the low communication and key distribution overhead associated with multicast group management.

Our approach is based on the Seraphim security architecture implementation [5], which uses active capabilities [6] for access control. Seraphim is an extensible, reconfigurable security architecture that is flexible and accommodates a wide variety of security policies and mechanisms. It also provides applications and users the ability to create dynamically and enforce highly customized and situational policies. Using these policies we have developed several secure multicast applications that demonstrate the flexible nature and low overheads associated with our architecture.

Keywords: *multicast, security, active networks, scalability, flexibility, policy, access control, active capability, reference monitor*

1. Introduction

Multicast is a useful network service that provides efficient, best-effort data delivery from a source to multiple recipients. The use of multicasting is becoming more and more widespread, as is demonstrated by the popularity of the experimental Mbone multicast service and its supporting applications. With video con-

ferencing via the Internet becoming extremely popular, multicast becomes an important technique to reduce sender transmission overhead, network bandwidth, and the latency observed at the receiver side. As multicast applications are widely deployed, the need to secure multicast communications becomes critical.

Providing security to multicast still remains a challenging problem due to the difficulties involved in securely distributing a session key. Traditional approaches involve a central key distribution center and these approaches do not scale well, especially when the multicast group members were scattered across a wide area network. These schemes rely on the existence of an asymmetric key infrastructure where every member who wishes to be part of the multicast network, needs to possess a private/public key pair. They also suggest that the new session key be unicast to each and every member, encrypted in the member's private key. The Core Based Tree (CBT) [4] and the Protocol Independent Multicast (PIM) [8] were proposed to address the issue of scalability, while papers like [7] introduced newer methods of key generation and distribution.

However, even in these approaches, one requires sender-specific keys to authenticate the sender of a message. New members must be given the session key and probably participate in a mutual authentication protocol. Also, all these methods require that the group (or session) key be changed when a particular member leaves the group, to prevent that user from eavesdropping on the rest of the traffic. Of particular concern here is the mechanisms used for distributing the new session key to the remaining users. The distribution overhead is usually very large [11]. This inefficiency makes traditional secure multicast applications very hard, if not impossible, to be used in very dynamic environment, which has frequent multicast joins and leaves.

In this paper, we propose a new approach for multicast security based on active networking technology and our dynamic policy framework. We use *reference monitors* at active routers and an *active capability* to

*This research is supported by DARPA F30602-98-1-0192

control the delivery of multicast packet. Active capabilities are essentially active capsules that are signed and concisely encode access control policy specifications and other security attributes associated with the underlying active protocol, e.g., multicast. Reference monitors provide a sandbox like environment on the routers and securely execute the active capabilities and enforce the policy specified in them. Using this approach we can flexibly support various secure multicast applications and reduce the communication overhead considerably.

Our paper is organized as follows. Section 2 of the paper gives a brief overview of our architecture and talks about its place in the general active network architecture. It also focuses on the implementation of the reference monitor and its flexible policy framework, which allows us to create and enforce dynamic policies. The next section talks about one particular type of access control policy, Role Based Access Control (RBAC) policy. The fourth part of the paper describes the dynamic secure multicast experiments in detail. The fifth part discusses the related work and the final part presents our conclusions.

2. Overview of Seraphim Architecture

An active network [15] provides a software framework that enables network applications to customize the processing of their data. Active applications inject capsules that contain programs (along with data) into the network. Active routers dynamically install these programs and execute them on the data. The basic software on an active router consists of three distinct, functionally separate layers: the application, the EE(Execution Environment), and the NodeOS. The NodeOS is similar to the kernel of traditional operating system. The EE is similar to a user shell. It provides an interface for accessing the NodeOS resources and an execution environment for application capsules. By using the shell provided by the EEs, applications create capsules with protocol code and/or data that can be installed dynamically in remote routers.

In order to provide security to the active networks infrastructure, and to provide dynamic, interoperable, and application customized security policy, we have implemented an architectural framework based on and built into the underlying active network architecture. The major components of our architecture and their interactions, in the context of the active network architecture are shown in Figure 1.

The security proxy component was added as a temporary module in our design, since the active network community is still working on the specifications of a standardized NodeOS interface [12]. The proxy acts as

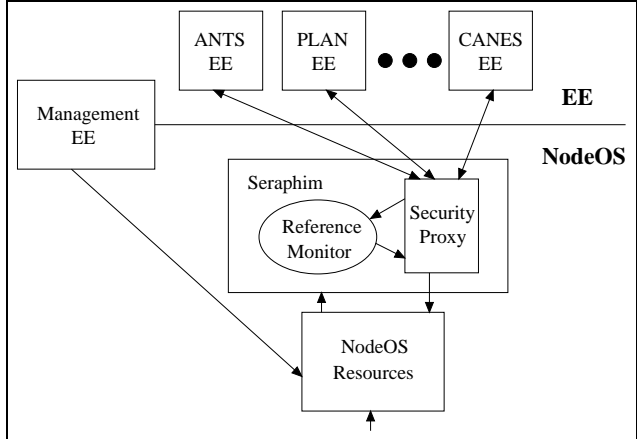


Figure 1. Secure Active Network Node

a wrapper to the NodeOS API and redirects the access requests to the reference monitor and then the evaluation results to applications.

2.1. Reference Monitor and Active Capability

The key component of our architecture is the reference monitor. The reference monitor is implemented as a co-located extension to the NodeOS. Every node has a reference monitor through which all accesses to the node resources occur. A dynamic, reconfigurable policy framework is developed as a component of the reference monitor for application customized policies. The policy framework itself is reconfigurable and it can be downloaded dynamically when required. Applications or administrators use the interface provided by the policy framework to create a customized piece of code that encodes the type of access control policy and other constraints used in the access control decision making process. This code fragment is called the *active capability* (AC) [10].

Unlike a traditional capability, which is merely a static authorization credential that encodes the principal and the permissions associated with the principal, an active capability is actually an executable Java script in our implementation. In addition, an active capability is protected by digital signatures, resides in user space, and can be freely passed around. Conceptually, an active capability is a piece of unforgeable code that encodes a critical, application-specific part of the decision making code used in access control. The active capability can encode various situational policies that depend on system attributes such as the current system time, resources, quota, etc.

An active capability relies on a policy framework for context. An application presents an active capability along with its regular data or protocol capsules to

the active router’s reference monitor at execution time. The access control policy type and user credentials are extracted from the capability. The remote router’s reference monitor recreates the context of the policy type within its policy framework. If at any point during this process, the policy framework discovers that it does not have an implementation for the type of the policy, it downloads the code dynamically into the framework, using the underlying active network. It then instantiates the run-time parameters associated with the application in its sandbox-like environment and executes the active capability in this environment. Based on the result of the evaluation of this active capability, the access control decision is enforced.

The principal of the active capability, which is typically an application user, must be authenticated by a trusted authority. The trusted authority also acts as the policy server in our system. This entity is responsible for generating and keeping track of the active capabilities. Usually, we associate one or more policy servers with each protection domain. Application programs contact their nearest or least-loaded server dynamically and obtain the active capability dynamically.

2.2. Policy Framework

In order to support ACs and provide users more flexibility in terms of policy specification, we have implemented an object-oriented policy representation framework in Java. This allows users and commercial organizations to specify policies tailored to their specific operational needs. The framework itself is a hierarchy of classes as shown in Figure 2.

The framework is dynamically configurable and extensible. The classes at the bottom of the framework are mostly abstract and are mainly used to represent mathematical concepts such as sets and mappings. These classes form the basis for a hierarchy of successively incremented specialized classes representing concepts such as labels and access control lists. Finally, at the top of the framework are classes which can be used to represent a variety of generic policy forms.

The policy framework supports the following common types of access control: Mandatory (MAC), Discretionary (DAC), Double Discretionary (DDAC), and Role-based (RBAC) [13]. More application specific access control policy systems can be easily extended from this object-oriented framework ([9] provides several good examples). In our model, we can specify not only the traditional $\langle \textit{subject}, \textit{object}, \textit{operation} \rangle$ access control triple, but also include a resource limit on usage, situational decision rules, constraints and dependences, e.g., based on current time of the day or current role of the principal.

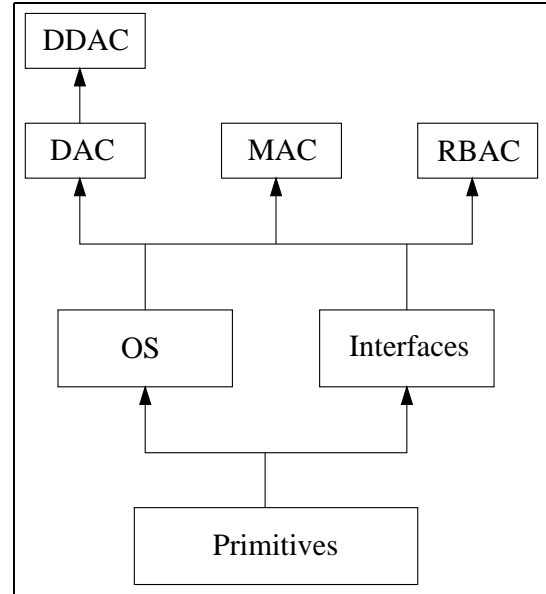


Figure 2. Component-level Map of the Policy Framework

Our framework also lets users specify pre-conditions and post-conditions. Pre-conditions allow necessary security checks to be performed before the actions take place, and post-conditions can be used to maintain state and perform additional checks after the action has been completed and when more information becomes available. An administrator GUI is provided as front end to the policy framework (Figure 3).

2.3. Dynamic Access Control

In order to apply our policy framework to active networks, we use active capabilities to distribute the permission information of the policies. We also componentize the framework so that it can be dynamically downloaded component by component. Active capabilities(AC) encapsulate a customized piece of code that encodes the type of access control policy and other constraints used in the access control decision. The framework uses the policy server as a communication front-end to accept AC requests either from reference monitors or applications and to provide the requested customized AC. Figure 3 shows the interaction of the various components of the policy administration mechanism. The policy administrator and the policy server are trusted entities.

There are three ways to manage the distribution of the ACs to the reference monitor:

- The applications can create and obtain application

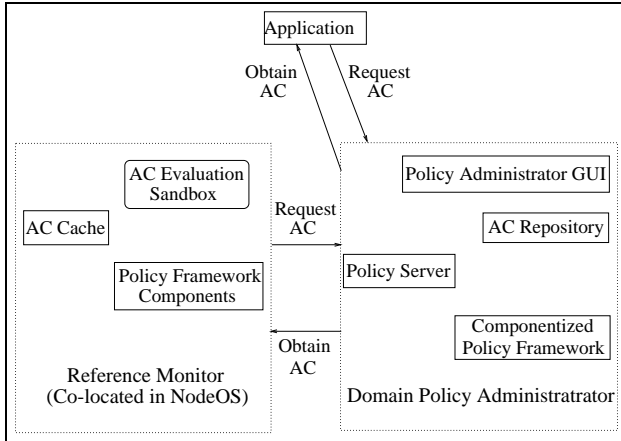


Figure 3. Policy Administration

specific ACs through the policy server GUI and send the AC along with active capsules. When a capsule arrives at a remote node, it is demultiplexed to the appropriate EE. The EE presents the AC to the security proxy along with its request to a NodeOS resource.

- If the application capsule does not have an AC, upon receiving a resource request via the EE, the reference monitor contacts the domain policy server. The policy server responds to this request with the appropriate customized AC.
- For common applications or frequent users, the policy server may distribute the ACs in advance to the reference monitors during system initialization.

Another important attribute of this architecture is the ability of the trusted authority, represented by the policy administrator or server, to revoke a capability at any point in time.

3. Role Based Access Control (RBAC) Policy

Our policy framework includes Role Based Access Control (RBAC) policy type, which is used in the secure multicasting applications presented in this paper. A Role Based Access Control policy, as the name suggests, uses the concept of a role as its basis for representing permissions [14]. It is a form of access control that emerges in the context of security policies for organizations. A role is chiefly a semantic construct that forms the basis for an access control policy. With RBAC, system administrators create roles according to the job functions performed in an organization, grant permissions to those roles, and then assign users to the

roles on the basis of their specific job responsibilities and qualifications. The idea is that the particular combination of users and permissions brought together by a role tends to change over time while the permissions associated with a role are themselves relatively more stable.

The biggest advantage that RBAC has over other forms of access control is that it is extremely intuitive to use and maps easily to real-world situations. A hierarchy of roles with senior roles inheriting all the permissions of junior roles closely follows the structure of organizations. The access control policy in RBAC is embodied in components such as role-permission, user-role and role-role relationships. These components collectively determine whether a particular user is allowed access to a particular operation on a particular component. These individual components can be easily (and intuitively) configured to provide the required degree of access control. For example, adding a new user to a system would merely involve assigning appropriate roles to the user according to the user's functions in the organization. Likewise, changing the nature of, for example, printer access, for all managers in an organization can be accomplished by merely changing the permissions with the manager role in the organization. All managers can immediately see the effects of the change.

RBAC is the most flexible type of access control policy. All RBAC subjects are assigned roles. Each role represents a particular set of objects and the allowed operations on each object. The major benefits of this aggregation are the considerable saving in terms of space and simplification in terms of management and enforcement. RBAC allows users to create policies with more sophisticated specifications than simple DAC, DDAC or MAC. A single user may have many different roles, and different permissions depending on the current role. Different constraints related to role and privilege may be enforced in RBAC. The constraints supported in our RBAC implementation include three important ones [13]:

- Mutually exclusive roles/permissions. This is the most common RBAC constraint. The same user can be assigned to at most one role in a mutually exclusive set. This ensures separation of duties.
- Prerequisite roles/permissions. If a role is declared a prerequisite for another role in the system, it means that a user may belong to the latter role only if the user already belongs to the first one.
- Cardinality constraints. This constraint imposes a limit on the number of users that can be assigned to a role. Similarly, this constraint can limit the

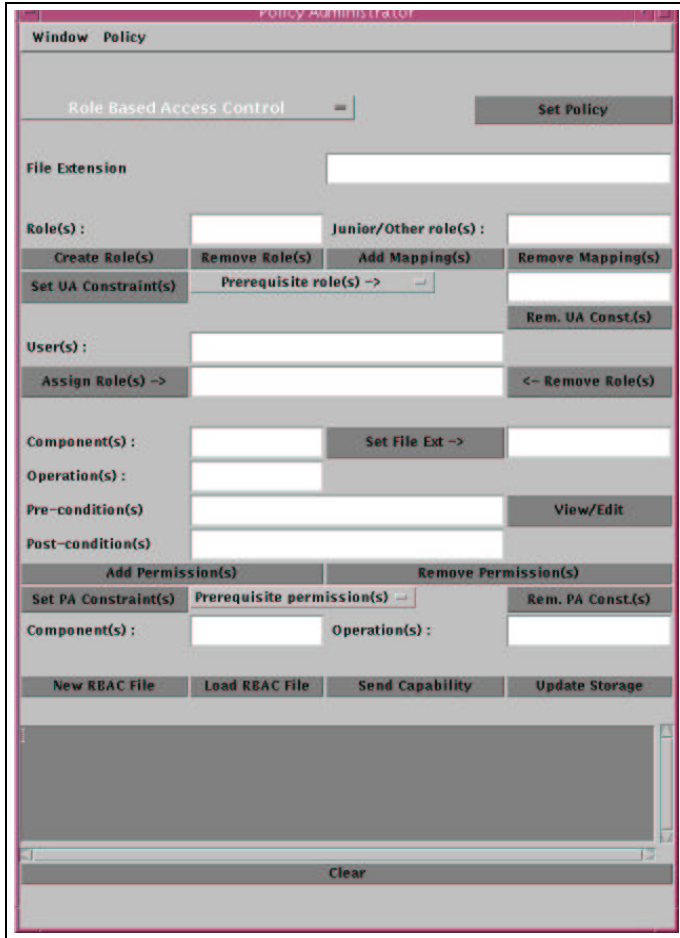


Figure 4. Policy Administrator GUI for RBAC

number of roles that a permission can be assigned to.

The policy administrator uses a GUI which allows users or system administrators to create and define policy specific attributes, and generate active capabilities. A screenshot of the RBAC GUI is shown in Figure 4. This GUI allows the administrator to create role definitions and associate users and permissions with the role, and supports other functionality (see [13] for more details).

4. Dynamic Secure Multicast

In this section, we are going to describe several dynamic multicast applications. The purpose is intended to showcase the benefits of using the RBAC policy class implementation and demonstrates the creation of dynamic multicast groups in active networks. In addition we also demonstrate a range of different policy specifications using active capabilities.

Our testbed implementation is based on the ANTS toolkit developed by MIT [15]. The original toolkit was written before the architecture group was formed, and did not reflect the layered architecture. Our first task was to split the design into layers and separate the functionality of the original Node class into distinct NodeOS and EE components. This modification is backward compatible and the original ANTS applications can run in our Secure Active Interoperable Network Toolkit System (SAINTS) [5].

4.1. The Multicast Tree Formation

While the ANTS implementation of multicast subscription was used for the multicast join, it provided no implementation for multicast leave. Modifications were made to the ANTS multicast subscription code, and the multicast unsubscription capsule was also written, to take care of dynamic multicast leave. This results in automatic construction or pruning of the multicast tree.

The multicast tree is dynamically formed and pruned by the generation of subscription and unsubscription capsules. The multicast subscription and unsubscription capsules are sent by a receiver towards the group owner (sender). These capsules dynamically construct and prune the multicast tree, respectively, when they are evaluated in the active nodes along the path from the receiver to the sender.

The Figure 5 show the steps of the multicast tree is formed when two receivers join a particular group in our implementation. The sender in the figure may refer to the Core in CBT [4] or the Rendezvous Point in PIM [8].

Similarly, when an unsubscription capsule is passed upwards, links are removed and the tree is pruned as shown in Figure 6, using active network features.

4.2. Pay-Per-View and Sneak Preview

As mentioned earlier, most of existing secure multicast schemes are based on shared session key among the subscriber nodes to ensure privacy of data. The main problem with this approach is the prohibitive overhead associated with the need to change the session keys every time a member leaves the group. In order to facilitate dynamic joins and leaves, our implementation uses active capabilities that grant and revoke access to the sensitive multicast data to the users in the multicast group. When a user decides to join, the trusted agent installs a capability that gives the user privilege to receive the multicast data, on the user's local node. The trusted agent may delegate this responsibility to the multicast source node or intermediate active routers.

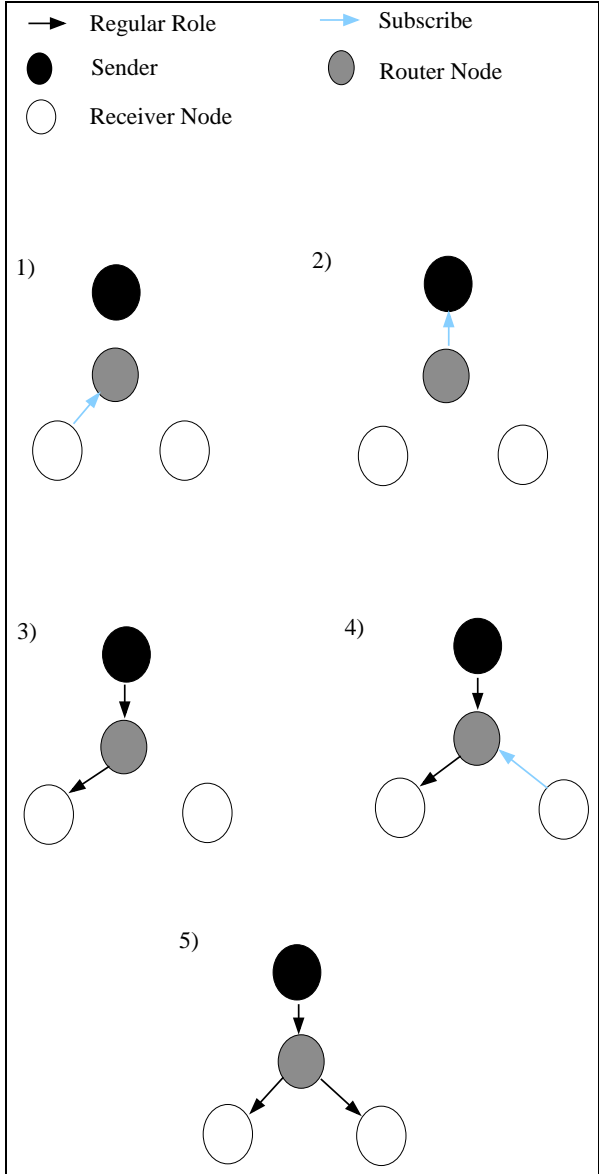


Figure 5. Multicast Subscription in Active Network

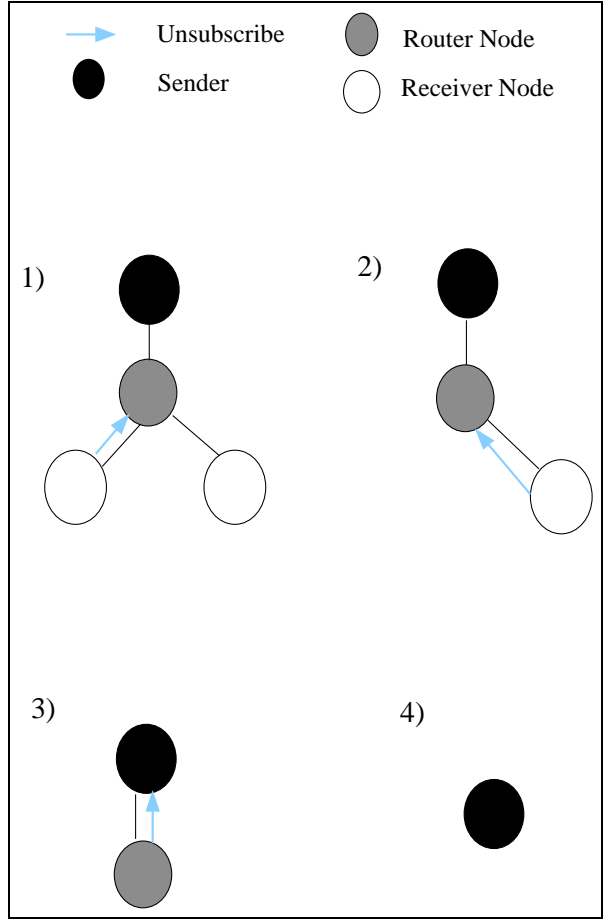


Figure 6. Multicast Unsubscription in Active Network

When the user leaves the group, the active capability for the user on the local node is revoked by the trusted agent. By doing that, we don't need to change any shared secret among the subscribers and thus can reduce the large session key distribution overhead. Our experiment scheme has a much lower overhead compared to the traditional schemes in place today.

Using our modified multicast application from ANTS, we devised two different experiment scenarios. In the first experiment or "Pay-Per-View", any user that wishes to receive a sequence of special multicast packets contacts the trusted agent and obtains an active capability that has a resource limit built into it. This capability is then installed in the reference monitor. Every time a special packet is delivered to the node, the resource limit is decremented by one. When the resource limit reaches zero, the active capability expires and the user can no longer receive the special multicast data traffic. If the user wishes to receive more, then the user "pays" and gets the another active capability with an appropriate resource limit.

The actual implementation was done using Role Based Access Control (RBAC). Users were assigned a default role (*Regular* role), that did not let them receive any of the special multicast data packets. Once they "paid", their role was replaced by a *Special* role that gave them the access rights for certain number of special multicast data packets (Figure 7). When the resource limit reached zero, the *Special* role was revoked and the default *Regular* role was resumed. A simple extension of this example is selective blocking. The sender can revoke the roles of selected nodes and implement dynamically changing selective blocking for different domains.

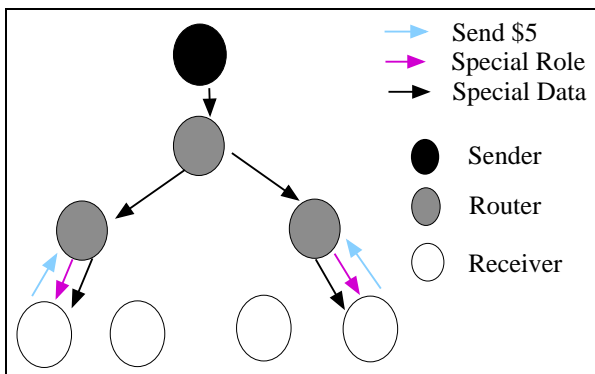


Figure 7. Pay-Per-View

The second demo used time-stamped active capabilities to control the access control decision. This experiment, or "Sneak-preview" allows any user in the multicast group to get some fixed time period (for example, two minutes) worth of free multicast data. Active ca-

pabilities are obtained and installed in the same way as in the first experiment. Once installed, the active capability keeps track of the local time and expires after two minutes have passed. The users may choose to receive any two minute period of the multicast traffic, all at once or in parts at any time. When the two minutes are used up, the active capability expires and user cannot receive multicast data anymore.

Both the "Pay-Per-View" and "Sneak-preview" experiments dramatically reduce the amount of state maintained by the server, compared to the state maintained by existing secure multicast mechanisms. And by eliminating need for using or changing session keys, these experiments distribute the server processing load among all the participating active routers and allow asynchronous, client-side initiated joining and leaving.

A possible extension of our applications is secure emergency multicasts. Emergency notification of events like a storm warning must be secure to be effective. "Geo-casting" scheme can be used to alert or warn subscribers about natural disasters like tornadoes passing through a geographic region. A multicast group is formed using dynamic join and leave, based on the geographic information. As the tornado moves, the multicast group can move along with it by adding and removing appropriate receivers in real time, using our fast join and leave. Active multicast data capsules can be sent to meteorological "subscribers" in a larger area, or to mobile users to warn that they are entering a danger area.

5. Related Work and Discussion

Currently active networks research community is applying active technology to multicast in the areas of congestion control [1] and error recovery [2]. We are integrating our Seraphim security system into them. We use our RBAC policy to control the signaling procedure of CANEs, and to dynamically control the installations of different protocols. We can also use our framework to control access of data packets dynamically.

We also use Seraphim to provides security service to an NS2 simulation [3] of a new secure multicast routing protocol. In the simulation, some multicast groups need higher security routing. The system gets the security information of the routers in the network, for a particular multicast group, based on their security clearance level. The simulation uses this information to set up proper secure multicast routing trees. The MAC policy of Seraphim is the access control policy for this application, which assigns different security clearance levels to routers. The Seraphim reference monitor functions as the enforcement engine and ensures that

the multicast joins follow the established security level hierarchy.

The overhead of our reference monitor and AC evaluation with proper cache scheme is negligible [10]. A efficient, distributed AC management is important for the overall system performance. Further research is required to design and implement such an AC management system.

6. Conclusions

This paper presents an alternative approach to secure multicasting. In the traditional approach when a user leaves a secure multicast group, the sender typically has to send messages to all the subscribers, in the entire multicast tree or subtree and change the session keys. Furthermore, during the transition period some of the sensitive information may be compromised. When a large number of users leave, and in some cases, rejoin, then this communication overhead, requiring reliable distribution¹ of new session and updating becomes prohibitive.

In our proposal, active capabilities carry the security information, using point-to-point secure channels. When a receiver leaves, an active capability changing the receiver's role is sent to the corresponding receiver router. Our solution does not rely on the multicast infrastructure for reliable delivery of an active capability. The reference monitor which is a co-located extension to our NodeOS kernel in active routers provides a safe, sandbox like environment for the execution of these active capabilities and dynamic enforcement of the policy associated with the active capabilities. The overheads associated with key distribution and scaling also dramatically diminish.

In addition, by embedding situational policies like pay-per-view, sneak-preview, selective blocking etc., we have demonstrated that our framework is flexible enough to satisfy a variety of secure multicast specifications. The dynamic nature of our security makes it very attractive for interactive sessions and video-conferencing.

References

- [1] CANEs Project Homepage
<http://www.cc.gatech.edu/projects/canes>.
- [2] PANAMA Project Homepage
<http://www.tascnets.com/panama/>.
- [3] UCSC Multicast Research Homepage
<http://www.cse.ucsc.edu/research/ccrg/>.

¹Reliable delivery in multicasting is in itself a hard problem and various solutions that have been proposed have high overheads associated with them

- [4] T. Ballardie, P. Francis, and J. Crowcroft. Core based trees: an architecture for scalable inter-domain multicast routing. In *Proceedings of the ACM SIGCOMM '93*, San Francisco, CA, September 1993.
- [5] Roy H. Campbell, Zhaoyu Liu, M. Dennis Mickunas, Prasad Naldurg, and Seung Yi. Seraphim: dynamic interoperable security architecture for active networks. In *IEEE OPENARCH 2000*, Tel-Aviv, Israel, March 26–27, 2000.
- [6] Roy H. Campbell and Tin Qian. Dynamic agent-based security architecture for mobile computers. In *the Second International Conference on Parallel and Distributed Computing and Networks*, Brisbane, Australia, December 1998.
- [7] G. Caronni, M. Waldvogel, D. Sun, and B. Plattner. Efficient security for large and dynamic multicast groups. In *Proceedings of the Seventh Workshop on Enabling Technologies, (WET ICE '98)*, IEEE Computer Society Press, 1998.
- [8] D. Estrin, D. Farinacci, et al. Protocol independent multicast – sparse mode (PIM-SM): protocol specifications. TETF draft, March 1997.
- [9] Tim Fraser. An object-oriented framework for security policy representation. Master's thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, December 1996.
- [10] Zhaoyu Liu, Prasad Naldurg, Seung Yi, Tin Qian, Roy H. Campbell, and M. Dennis Mickunas. An agent based architecture for supporting application level security. In *the DARPA Information Survivability Conference and Exposition*, Hilton Head Island, SC, January 25-27, 2000.
- [11] Suvo Mittra. Iolus: a framework for scalable secure multicasting. In *Proceedings of the ACM SIGCOMM '97*, Cannes, France, September 1997.
- [12] L. Paterson et al. NodeOS interface specifications. AN NodeOS Working Group, Draft, 1999.
- [13] Vijay Raghavan. On the design and implementation of a security policy administration for a dynamic security system. Master's thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, May 1999.
- [14] R. S. Sandhu and E. J. Coyne. Role-based access control models. *IEEE Computer*, 29(2), February 1996.
- [15] D. Wetherall, J. Guttag, and D. Tennenhouse. ANTS: a toolkit for building and dynamically deploying network protocols. In *IEEE OPENARCH'98*, San Francisco, CA, April 1998.