A Flexible and Secure Multicast Architecture for ATM Networks

Shaw-Cheng Chuang

21st December 1994

Abstract

We describe our unifying architecture for multipoint-to-multipoint communications in ATM networks which meets the diverse requirement of group communications and permit a large degree of control on the multicast group. With an integrated use of a name space, the scheme allows scalable extension to large scale wide area multicast communications. We also enable flexible control on routeing architecture and group membership. We consider security service to be a fundamental element of the multicast communication. A framework of multicast authority and multicast coordination centre is used to provide the dynamic group management and membership support, with a integrated security mechanism for flexible group access.

Implementation experience of this architecture on the XUNET wide area ATM testbed will also be discussed.

1 Introduction

Multicast communication is an increasingly important service in today's networks. For ATM network, the ATM Forum has defined a rather rudimentary point-to-multipoint signalling support for multicast service[1]. There is however a whole range of other important issues to be addressed before a general-purpose multicast service is possible. A close examination of multicast applications indicate that a more adaptable architecture is required. This paper presents a flexible and secure architecture for *multipoint-to-multipoint* communications in ATM networks. In this paper we tackle, in a integrated approach, issues such as:

- The service model and group control
- Multicast group naming and management
- Dynamic membership support
- Routeing architecture
- Access control
- Signalling support
- Application Programming Interface

Multicast architecture in the past tends to have a partial handling of the above issues. Our key contribution is in the provision of a unifying architecture for multipoint-to-multipoint communications which meets the diverse requirements of group communications and permits a large degree of control on the multicast group. We have taken a pragmatic view in our approach and ease of implementation is one of our primary aim. We have an integrated use of name space. We also enable flexible control on routeing architecture and group membership. We consider security service to be a fundamental element of the multicast communication. A framework of multicast authority and multicast coordination centre is used to provide the dynamic group management and membership support, with a integrated security mechanism for flexible group access.

Implementation experience of this architecture on the XUNET wide area ATM testbed [2] will also be discussed.

2 Service Model and Group Control

Before the multipoint communication architecture can be defined, we need to define the service model. The service model defines what the service looks like to users of the service interface. We define a multicast group to be a collection of communication endpoints with different communication capabilities (e.g. transmit, receive etc) identified by a *group name*. A multicast call is an instantiation of communication between the endpoints in the group, where data sent by one endpoint is received by receive-capable endpoints.

A multicast group is conventional identified by a *group address*. We do however think this is a rather inappropriate representation. An address is most frequently used to aid the location of objects. In a unicast call, the address of the peer party is typically used in the routeing process for locating the peer in the network. In the context of a multicast group call, it can be argued that the *address* of a group does not serve any meaningful purpose. One question to ask ourselves is *"What is a location of a group ?"*. A multicast group by our definition is a collection of network endpoints, each with either transmitting or receiving capability. Thus the concept of a location of a group does not make sense. We can therefore argue that the identifier of a multicast group is a name rather than an address. We have put more emphasis on the name of a group rather than the address since we believe that we can utilise the structure of the name and it's registration in the directory system to provide a mechanism for group management.

The group name in the structure of an Internet DNS Domain Name [3] or a Distinguished Name in the CCITT X.500 [4] system enables the multicast group to be registered in the directory system. This entry in the directory system provides a convenient means to identify the multicast service and locate the appropriate *Multicast Co-ordination Centre* (MC^2) (see below). The directory is used both to locate the management entity of the multicast group and also a repository of certificates that is used to verify the multicast group entry and authority. The management entity (MC^2) of the group is the enforcer of group policies and manages the group's access control.

The registration of the group entry in the directory and the creation of the MC^2 is authorised by a *Multicast Authority* (MA). Each MA is responsible for a particular *Multicast Authority Domain*. The MAs for the different multicast domains form a trusted hierarchy which is vital to the construction of groups and access control to a group using the *MembershipCard* protocol described later in this paper.

We have the notion of the *owner* of a group who is the creator of the call and serves the role of *policy maker*. Responsibilities of the owner includes quality of service selection, accessibility, notification, connection permissions, group termination etc. The owner need not be a member of the multicast group and ownership can be transferred during the lifetime of the call. The *owner* of a group interacts with the MA to request permission for group creation.

3 Dynamic membership support and control

A careful consideration of a multicast service model will reveal that it consists of two different components, the data path and the control path. The data path deals with information distribution to the members of the multicast group, i.e. the communication channel. The control path handles the setting up and tearing down of the multicast data path, ownership of the group, admission and deletion of members to the group etc. The control path function is accessible by the control interface. In our architecture, the interface is defined using the *Interface Definition Language*. Distributed processing technologies such as Remote Procedure Call (RPC) is used for all control mechanisms in our architecture. This is in alignment with work within the TINA [5] initiative to define a telecommunications information networking architecture based on advanced distributed processing technologies.

The membership of a group can be either static or dynamic. Dynamic membership allows members to join and leave during the lifetime of the group. The virtual circuit nature of ATM requires a connection setup stage before any data transfer. This presents a challenge to the management of multipoint-to-multipoint communications. It is critical in multipoint-to-multipoint communications that the endpoints can obtain the correct instantaneous membership information for connection requests to be launched. In our architecture, the *Multicast Co-ordination Centre* MC^2 plays a key role in the maintenance of dynamic group membership. Indeed, the MC^2 is the command and control centre for all operations of the multicast call and therefore forms the centre piece of our architecture. The MC^2 provides an interface that allows atomic updates/retrieval of membership information.

Our multicast service model for ATM networks provides a wide range of control. As we have discussed above, the group owner controls the group policies and holds responsibility for the group. Besides owner control, the control service model also consists of control from endpoints (acceptance/rejection of call, capabilities etc) and the network. Each endpoint also exports an interface, to be used by the MC^2, for both the notification of the multicast call state and detection endpoint liveness. For instance, the group policy might be to notify current endpoints when a new group member joins. The network also has the control over the multicast group, using the *MembershipCard* access control protocol and the call admission control mechanism for call/traffic control. As we shall see later this paper, the *MembershipCard* also enables policy-based routeing for the establishment of group join.

4 Routeing architecture

One main objective of multicast communication is to reduce the cost of the data path, i.e. common path in the data distribution is grouped so that only one copy of the data is sent. The formation of this data distribution path is a routeing problem. A choice of routeing architecture is however a tradeoff between the ease of group management/control and data cost. Different routeing architectures such as shared tree or source-based tree each have their merit. Recent developments in the Internet community on the PIM[6] multicast architecture have further emphasised the importance of support for both tree types. One important characteristic of our multicast architecture is the support of both the shared tree and source-based tree, with the use of the MC^2. A multipoint-to-multipoint group can either be constructed from a single shared-tree or a mesh of source-based trees. The MC^2 implements several functions in this respect:

- Enables the owner to specify the preferred tree type at group creation time.
- Provides information on tree type to the endpoints so that the appropriate joining procedures can be taken.
- Solves the problem of locating the multicast server or the core in a shared-tree. This is an area that is ignored by the Internet community.
- Provides a consistent set of membership information to all endpoints for the purpose of establishing virtual circuits.

We have also identified Naive Routeing [7] to be the prime candidate for multicast routeing due to its properties of simplicity and easy integration with unicast routeing with acceptable performance.

5 Access Control

Access control is an area that has been very much ignored in conventional multicast mechanisms, such as IP multicast. The access control mechanism in our architecture includes three areas:

- Access control to group information in the directory system

  The particular mechanism is dependent on the directory system employed and is not specified in this paper.

- Access control to group creation

  This is controlled by the Multicast Authority (MA). The MA holds responsibility for the multicast network resource control and a federation of MAs forms the trust structure for multicast access control. The scope of control of a MA is determined by local administration decision. Typically this will be coincide with domain's naming structure. The MA exports a control interface. Upon authenticating the owner of a group and with group creation request being authorised (the authorisation could take the form of an access control list), the MA grants the creation of new group and updates the directory system.

  In a simplified and less-automated environment, the MA could take the form of a system administrator who makes policy decisions and updates the directory manually.

- Access to a multicast group

  This includes several types of access:

  – Access to the MC^2

    The MC^2 contains a wealth of state information about a particular instance of a multicast group, e.g. receiver, transmitter, multicast server addresses etc. Restricting access to such information will reduce the risk of attack on the group. Again, an access control list can be used. The owner of the group has the "super-user" access to such information.

  – Joining a group

    The owner of a group sets the policies on membership at the time of group creation, using the control interface to the MC^2. Before an endpoint joins a group, it initiates a call to the MC^2. If the endpoint is authorised to join the group, a *MembershipCard* is issued, together with the current membership information required to join the group.

  – Network group authorisation

    The network not only needs to protect itself from unauthorised access to a group from an endpoint, it is also the responsibility of the network to prevent a seemingly legitimate access to an illegal group. During a group join, the endpoint needs to initiate network signalling to attach itself to the current group, either using Source Initiated Join or Leaf Initiated Join. In both cases, the *MembershipCard* is presented to the network, as part of the signalling message information element. The network will only complete the signalling process if the *MembershipCard* is valid and conform for network access policy.

  – Endpoints

    The Source Initiated or Leaf Initiated Join request will be further filtered by the receiving endpoint, be it the transmitter or receiver.

5.1 The *MembershipCard* Protocol

The *MembershipCard* protocol provides an off-line access control mechanism on a multicast group. A *MembershipCard* is a cryptographically sealed access control token issue by the MC^2. It provides not only the certification for an endpoint to access a group and its rights (e.g. transmit, receive) in this group, but also the validity of the group within the multicast authority.

The implementation of the *MembershipCard* protocol relies on a framework for the authorisation of multicast groups, the *Multicast Authority* (MA). Similar to the X.509[8] Certificate Authority, the MA is certificate based. The trusted multicast authority assigns a unique name to each group and it manages two types of certificates:

- Multicast Authority Certificate

  This certificate provides the network with the ability to verify the *Multicast Authority Domains*. The MD hierarchy provides mutual certification and therefore we can always trace through the *Multicast Authorisation Path* to verify a foreign MA. An example form of the Multicast Authority Certificate is shown below, expressed in ASN.1[9]:

  ```
  Validity ::= SEQUENCE (
          notBefore       GeneralizedTime,
          notAfter        GeneralizedTime)

  MACertificate ::= SIGNED SEQUENCE (
      signature   AlgorithmIdentifier,
      issuer      Name,
      validity    Validity,
      MA          Name)
  ```

  A MA is typically run by the Network Operation Centre (NOC). A Multicast Authority (MA) signs all certificate. If Alice is in the same MA domain as the multicast authority certificate she would like to verify, this is easy since Alice knows the public key of the local MA. If they use different MAs, this is more complicated. The MAs certified each other MAs in a tree structure, just like the Public key Certificate Authorities. Thus cross domain *MACertificate* verification can be carried out by tracing the multicast authority certification path

- Multicast Group Certificate

  This is the certificate that a MA issues upon authorising the creation of a group. Upon the request of the owner of the group, the MC^2 is instantiated and it retains a copy of the Multicast Group Certificate, *MGCertificate*. The presence of this certificate indicates that the group is legitimately created under the authority of the issuing MA. An ASN.1 representation of the Multicast Group Certificate looks like the following:

  ```
  MGCertificate ::= SIGNED SEQUENCE (
      signature   AlgorithmIdentifier,
      issuer      Name,
      validity    Validity,
      group       Name,
      owner       Name,
      mc2Sap      PresentationAddress,
      charter     BIT STRING)
  ```

Both of these certificates are reposited in a directory system. We thus assume the existence of a directory infrastructure that cross multiple domains, e.g. DNS or X.500. X.500 might be a more suitable candidate as it allows dynamic modification of the data entries. A *MembershipCard* is then constructed based on these certificates.

MembershipCard is a cryptographically sealed message that can:

- certified one's membership in a group
- specified the rights of the member

- Identified and certified the issuing body
- Identified the multicast authority

When Alice wants to join a group, Grover, and has obtained a *MembershipCard* from Grover's MC^2. Assume Alice's MA is Austin and her management agent MC^2 is Manning. The *MembershipCard* is presented to the network, e.g. to initiate a connection from a transmitter, from Alice to a receiver, Carol. Bob, a signalling entity on the path from Alice to Carol, would like to verify the *MembershipCard*. Assuming that Bob's MA is Barham.
We have

```
Grover_MGScert      MGCertificate
Austin_cert         certificate
Austin_MAcert       MACerticate
Manning_cert        certificate
```

and a *MembershipCard* looks like:

```
MembershipCard ::= SIGNED SEQUENCE (
    signature           AlgorithmIdentifier,
    Alice_name          Name,
    Alice_sap           PresentationAddress,
    Manning_name        Name,
    Grover              Name,
    Austin              Name,
    rights              BIT STRING,
    lifetime            Validity)
```

With a push model of certificate management, the above set of certificates (Grover_MGScert, Austin_cert, Austin_MAcert, Manning_cert) will be sent along the *MembershipCard*.

There are two phases to the protocol: Obtaining the *MembershipCard* by Alice and verification of Alice's *MembershipCard* by Bob.

- Obtaining the *MembershipCard*

  There are two cases to consider:

  – Alice is in the same MA domain as Manning

    1. Alice can then verify the MGCertificate for Grover in the directory and contact Grover's MC^2, Manning.
    2. Alice also pick up the public key certificate of Manning.
    3. After the MC^2 has authenticated Alice and if Alice is entitled to certain right for the group, Alice's membership card is issued. Otherwise Alice get a refusal notice.

  – Alice is in different MA domain as Manning. Alice needs to get the "multicast authorisation path" from Alice's MA domain to Manning's MA domain. This will then be used to verify the MACertificate and MGCertificate. Then do step 2) and 3) above.

- Verifying the MembershipCard

  1. Bob obtains the "multicast authorisation path" from Barham to Austin. Austin_MAcert is then verified using Austin_cert and the "multicast authorisation path". This step verifies that Austin is a valid authority within our multicast authority framework.
  2. Grover_MGScert is then verified using Austin_cert. This step verifies that Grover is a valid group created under the authority of Austin. The validity of Grover_MGScert is also checked to make sure the group is still valid.

3. The lifetime of the MembershipCard is also checked.
4. Alice_name, Alice_sap, rights, Grover are used by the signalling entity for the management of the multicast call.

One of the key advantages of such system is the passive nature of the checking. The verification can be done at any stage in the network. In a local network environment where network resources accounting is not important, the checking of the *MembershipCard* can be waived. However, when crossing a multicast authority domain, such checking might be mandatory. The checking also does not require direct interaction between the network and the multicast entity, since all additional information required to verify the *MembershipCard* is stored in infrastructure like the directory system. When in a remote domain, the verification could still be easily achieved using the information reposited in the directory infrastructure, similar to verifying a public-key certificate. The decoupling of network access control verification from the MC^2 not only provides a performance gain (the network need not go back to the MC^2 for each verification), it also enables the MC^2 to freely evolve to meet different requirements, while still providing general access control. The performance of the protocol can be further enhanced by caching the keys used in the verification in the signalling entity. Therefore verifying further members of the same group has very little overhead.

The lifetime of the *MembershipCard* is kept short (a few minutes; this should be more than enough to complete a multicast call) so that there is no need for any certificate revocation mechanism.

Since the identity of the MA and the name of a group is sealed in the *MembershipCard*, they can be used in the process of policy route computation. Such terms can be easily integrated with the *Policy Terms* [10] used in policy routeing. Furthermore, with the use of Naive routeing, we can have a clean integration of policy routeing in multicast with those of unicast communications.

6 Signalling Support

The requirement for signalling support in this ATM multicast architecture includes Source-Initiated Join, such as those specified in ATM Forum UNI [1], and Leaf-Initiated Join (LIJ) as suggested in [11]. In addition, we also require an authenticated virtual channel for the purpose of communicating with the MC^2, MA, and signalling channel.

Depending on the level of signalling support, our multicast architecture can be implemented either "inside" or "outside" the network. When we refer to an "outside" network implementation, we are considering the situation where the signalling entities at the endpoints interact directly with the MC^2. We thus have the convenience of implementing an overlay multipoint-to-multipoint service over a rudimentary UNI service.

For value-added network service providers, they might be equipped with a more advance UNI protocol such as CMAP [12] where a true multipoint-to-multipoint abstraction is presented across the UNI, our architecture could then be implemented with total transparency to the endpoints. In this case, the interaction with the MC^2 is carried out with the switch signalling entities "inside" the network.

7 Implementation

An early implementation of this architecture was carried out on the XUNET wide area ATM testbed[2] during summer 1994. A prototype MC^2, with dynamic membership control, a rudimentary access control capabilities and support of the routeing architecture was built. Multicast extensions for Source-Initiated Join and Leaf-Initiated Join were also added to the XUNET signalling protocol. The application programming interface defined in section 7.1 was also implemented.

An indicated earlier, we used distributed processing technologies in the definition and construct of the architecture. We used the ANSAware [13] in our implementation. Interfaces to services such as that to the MC^2 are defined in the ANSA Interface Definition Language (IDL).

7.1 Application Programming Interface

The service model above only described the kind of service that a user (application) should expect. It however does not defined the user interface to such a service.

On the XUNET implementation, the application programming interface is an extension of those defined in [14].

7.1.1 open_connection

```
int open_connection(char *dstaddr, char *service, char *qos, int port, char *gp_name)
```

This is an operation performed by the source/core to initiate a call, either unicast or multicast. In both cases, the service name recorded in the MC^2 is formed by concatenating the fields *gp_name, port*. The *service* field indicates the group service name.

7.1.2 exportService

```
COOKIE_T *exportService(char *name, int port, char *gp_name, char *qos)
```

This is called by a server/receiver to export a service. The connect request from signalling entity will eventually be on the 'port' argument.

If *gp_name* == NULL, this is a for incoming call. This might be for both unicast or source-initiated add.

If *gp_name* != NULL, the receiver is registered with the MC^2 and LIJ is sent towards all the sources registered with the MC^2. *listen_connection* and *accept_connection* is then used to complete the call.

7.1.3 Connection control

```
int listen_connection(int my_sock, char *dstaddr, char *service)
int accept_connection(int my_sock, char *dstaddr, char *service)
int reject_connection(int my_sock, char *dstaddr, char *service)
```

This is called by both source and receiver to accept a connection request. The endpoint should create a socket on which to be contacted.

listen_connection blocks until a call request (either LIJ request or connection request) arrives. We can the use *accept_connection* and *reject_connection* to either accept or reject the call.

accept_connection returns VCI on success and error on failure.

reject_connection is also used by the source to close down selected receiver. A close on the sending socket by the source is equivalent to *reject_connection* on all the receivers and removing the source from MC^2.

7.1.4 Owner operations

```
COOKIE_T *create_mcast_group(char *group_name, char *qos, int port, int accessibility)
/* port - port owner listen on */
```

Owner creates a multicast group and listening on *port*.

```
int destroy_mcast_group(COOKIE_T *gp_id, int port)
```

Group destruction

```
int register_src(COOKIE_T *gp_id, int len, EP *eps, int port)
int delete_src(COOKIE_T *gp_id, int len, EP *eps, int port)
```

Registering and de-registering sources

```
int register_sink(COOKIE_T *gp_id, int len, EP *eps, int port)
int delete_sink(COOKIE_T *gp_id, int len, EP *eps, int port)
```

Registering and de-registering receivers

8 Conclusion

In this paper, we have presented the architecture of an ATM multipoint-to-multipoint service and a prototype implementation. We have shown that by employing a multicast coordination centre, we are able to exert control on dynamic membership, group access and flexible routeing scheme . Furthermore with an integration of naming scheme with directory registration, we are able to established user friendliness in terms of group identification, location of appropriate multicast coordination centre and a trusted framework for access control. *MembershipCard* protocol allows both easy passive checking of access control and integration of policy multicast routeing. We believe our unique approach of seamless integration of naming, dynamic membership, access control and routing architecture facilitates a flexible multicast service that meets the requirement of the applications.

References

[1] ATM Forum, "ATM Forum User Network Interface Specification V3.1," 1994.

[2] A. G. Fraser *et al.*, "XUNET II - A Nationwide Testbed in High Speed Networking," in *Proc. Infocom 92*, May 1992.

[3] P. Mockapetris, "Domain names - concept and facilities," November 1987.

[4] "The Directory – Overview of concepts, models and services." International Telegraph and Telephone Consultative Committee, 1988. CCITT X.500 Series Recommandations.

[5] W. J. Barr, T. Boyd, and Y. Inoue, "The TINA Initiative," *IEEE Communications Magazine*, 1993.

[6] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "Protocol Independent Multicast (PIM): Motivation and Architecture," *Proc. ACM Sigcomm*, August 1994.

[7] M. Doar and I. Leslie, "How bad is naive multicast routing ?," in *Procceding of Infocomm 93: Proceedings of the Tenth Annual Joint Conference of the IEEE Computer and Communications Societies*, 1993.

[8] "The Directory – Authentication Framework." International Telegraph and Telephone Consultative Committee, 1988. CCITT X.500 Series Recommandations.

[9] "Information Processing — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1)." International Organization for Standardization and International Electrotechnical Committee, 1987. International Standard 8824.

[10] D. Estrin, M. Steenstrup, and G. Tsudik, "A protocol for route establishment and packet forwarding across multidomain internets," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 56–70, February 1993.

[11] P. F. R. Bubenik, "Leaf Initiated Join Extension," *Tech Rep. ATM Forum/94/0325R1, ATM Forum Technical Committee, Signalling Subworking Group*, July 1994.

[12] M. E. G. J. D. DeHart and R. G. Bubenik, "Connection Management Access Protocol (CMAP) Specification," Tech. Rep. 92-01, Washington University, Department of Computer Science, January 1992.

[13] Architecture Projects Management, *ANSAware Version 4.0 Manual Set*, 1992.

[14] R. Sharma and S. Keshav, "Signalling and operating system support for native-mode atm applications," *ACM Sigcomm*, August 1994.