Managing concurrent QoS assured Multicast sessions using a Programmable Network Architecture

Radu State, Olivier Festor, Emmanuel Nataf

Managing concurrent QoS assured Multicast sessions using a Programmable Network Architecture

Radu State, Emmanuel Nataf, and Olivier Festor

LORIA - INRIA Lorraine - Université de Nancy II
615 rue du Jardin Botanique
F-54602 Villers-les-Nancy Cedex
France

Abstract. In this paper we address the management of concurrent multicast sessions with security and QoS guarantees. Their main feature is a high degree of change in terms of membership, implying the necessity for fast reconfiguration and provisioning. We will approach the management problem using techniques developed in the context of virtual private networks, adapted to the high dynamicity that we are confronted with. We propose a framework for the management of such networks by integrating the management and the control plane, using the programmable and active networks paradigms developed within the research community. We apply the framework to the management of residential user TV multicast, where an ATM based access network supports the delivery of TV content to all clients having subscribed to a DVPN.

Keywords: Active Network, P1520, VPN, multicast

1 Introduction

The advent of broadband technologies for the local loop, combined with the deregulation of this part of the network in most European countries, fosters the deployment of new services to the end user. One of these services is the multicasting of digital TV channels to all subscribed residential customers. Such a service requires both a dedicated signalling plane (e.g. for channel selection by the end-user) and a high performance management plane for provisioning, monitoring and flow management.

The project aims at providing a management framework for concurrent QoS assured multicast sessions in the backbone (here a metropolitan area network) to provision the local loop. Each TV channel is conceptually defined as a multicast tree which has the characteristics of a VPN, ie. Closed User Group, security and QoS guarantees. The support of multicast facility within VPNs is required in order to optimize the use of network resources. Since the required multicast trees are strongly dynamic, their configuration within the traditional VPN management time-scale is no more viable. Our work proposes an integration of the

management and signalling planes using programmable and active technologies in order to cope with this strong variability and dynamics.

To present the major components of our architecture, the remainder of the paper is organized as follows. Section 2 gives a definition of Dynamic Virtual Private Networks, illustrates the target backbone to be managed and motivates the need for a programmable architecture in order to enable in-time management. Section 3 and 4 detail the components of our management architecture. In section 5 we detail the distributed software architecture. Section 6 summarizes the work done in other projects which has been partially reused in our approach. Finally a short conclusion is given and future work is outlined.

2 Dynamic Virtual Private Networks

Dynamic Virtual Private Networks are from a logical point of view virtual private networks which encapsulate multicast trees whose topology may change very frequently depending on user interactions (join, leave a channel). These DVPN rely on Service Level Agreements (SLA) established between individual end-user subscribers and the service provider as well as on SLAs between the latter and content providers. The service provider may itself rely on a transport provider with specific SLAs. At the lowest level, the network is composed of ATM switches. At the backbone edges, customers access the dynamic virtual network through Service Access Points (SAP). The physical infrastructure uses the Asymmetric Digital Subscriber Line (ADSL) technology.

Each TV channel is modeled as a DVPN. This choice is appropriate since common features which are particular to VPNs (e.g: Closed User Group, Security and QoS guarantees) are of a crucial importance in the context of TV residential broadcast.

Traditional Management time-scale for VPN provision is not appropriate for such DVPNs. In fact, the topology change of each tree must occur in a couple of milliseconds whenever a customer zaps from one channel to another. Moreover, this configuration task must be performed in parallel enabling multiple customers to change channels simultaneously. Since generic network layer support is unlikely to support such requirements, part of the management tasks must be integrated with the signalling facility into a programmable infrastructure. Time constraints are one motivation, information sharing provides a second one. Both camps benefit: like in active networking, signalling protocols may benefit from information provided by the management framework (like topology or link states), and on the other hand, a management framework may benefit from information provided by the value added services signalling plane (e.g. actual number of customers who are looking at a given channel).

3 The management architecture

In order to enable management of those DVPNs, we have chosen to combine the programmable network approach and active technology, as detailed in this section.

As illustrated in figure 1, the management architecture is built within a CORBA DPE on top of P1520 [3,2] abstract switch interfaces. The entire architecture is composed of five elements.

The main component is the DVPN Tree Manager. This entity is responsible for the configuration, extension and reduction of DVPN trees on the backbone. It maintains a view of the topology of the physical network as well as a logical one for each each DVPN currently in activity. This entity offers an API to the Customer Service Access Point (second component) through which all channel setup/change requests are received, acknowledged (check that the user is allowed to join a given DVPN) and performed (issue an expansion request to the DVPN for the given SAP and the given channel.

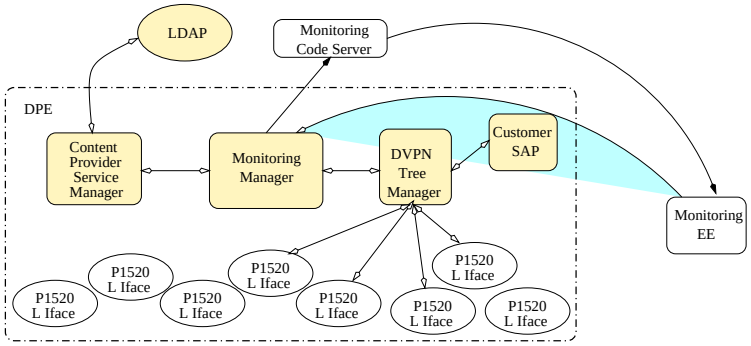

Fig. 1. The management architecture building blocks

The third component is a monitoring entity. This management entity provides facilities for deployment of monitoring code for both the Service Provider and the Content Providers. Related to the monitoring entity is the Monitoring EE, an Execution Environment for active code. This EE is availaible in various nodes of the network, especially on each edge node as one special End-User. This EE is used by the Monitoring manager to deploy service specific monitoring code for one or multiple content provider service parameters. This monitoring code is developed by the network management staff according to SLAs. The resulting code can be dynamically downloaded and controlled by the Monitoring Manager in accordance with the service management entity described below. The functionality that we achieve with this approach corresponds to a on-the-fly construction of RMON Mibs. For usual traffic monitoring, the monitoring manager uses SNMP-based agents located in the switches, at least in the first release

of the management environment. The monitoring manager relies on a monitoring code server offering a set of monitoring functions which can be deployed to the probe EEs onto the network.

The last component is a service level manager called Content Provider Service Manager, responsible for both DVPN setup and collection of data that must be made available to the content provider (mainly based on what is specified in the Service Level Agreement. This element mainly relies on the monitoring manager to gather information from the network. A second information source is the Customer SAP through which coverage related data can be obtained. The service level reports are made avalaible to the content provider through an LDAP directory server.

All those components are currently under development using the Java technology and a CORBA DPE at the programmable level. The Execution Environment and active code is an extension of the ANTS Toolkit [12] running on a Linux box for probes.

4 DVPN Tree Manager

The DVPN Tree Manager is the core of the management platform.We will first describe the static information model, that is the structure of the information needed to maintain a view on the DVPN and public network properties. Afterwards we will continue with the functional aspects of its activity.

4.1 Information Model

The information needed in order to perform the management is twofold, since two different layers of abstraction can be put into evidence. The first one concerns the public network used to deliver the DVPN service. The underlying public network is modeled using a slightly modified information model of the one introduced in [10]. The extensions to that model concern the support of multicast connections at a layer trail and respectively subnetwork connection layer. At the DVPN level, new entities are introduced and linked to the supporting elements from the public network layer view (see figure 2). For the sake of clarity we keep the public network part of the information model quite simple and generic. Specific ATM related entities are obtained by a specialisation of the generic classes. For instance :

- the `LayerTrail` class can be derived in order to stand for a ATM SVC. It has several ATM related properties in terms of traffic descriptors, QoS service class and parameters.
- By a specialisation of the `Subnetwork` class, one can model physical ATM switches. However, this class can also model a collection of ATM switches performing as one global switch.
- `SNC` (Subnetwork Connection) represents physical cross-connects done in the switches, but also communication across a collection of switches, considered

to jointly form a `Subnetwork` entity. It is worth noting that the `SNC` object has several destination endpoints and one designated source endpoint in order to perform at switch level the point to multipoint cross-connections.
- `nwCTP` models connection endpoints. In the case of a ATM residential backbone, it is mapped to a triplet (port, VPI, VCI) representing the interface and the connection identifiers for the particular connection.
- `Link` represents the connectivity at the physical layer.
- `LinkTP` models the termination point of a `Link` object and is mapped to the particular switch interface used.

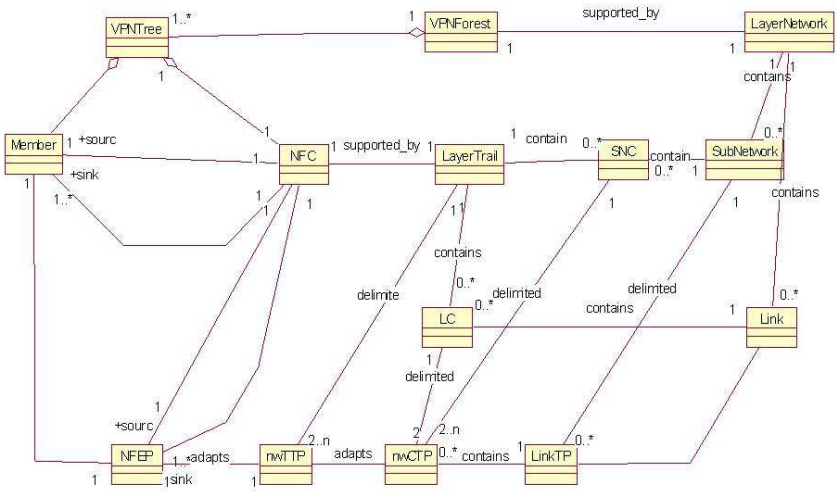

Fig. 2. DVPN Information Model

The information model for VPNs, that we extended for reuse in the context of DVPN, has been introduced in [8]. We enhance the model in order to be able to:

- model a collection of DVPNs as seen by a global view. The class VPNForest is introduced for this purpose.
- represent the Multicast Tree that corresponds to a particular DVPN. At the DVPN level, such a tree is a collection of network flow endpoints (`NFEP`) and the traffic from one source `NFEP` to the remaining ones. This traffic is modeled

using the network flow connection class (NFC) and represents the video data delivered to the end users. The NFC is a one-to-multipoint connection in order to model a multicast tree. It is characterized by a series of QoS parameters. If MPEG format is used, these parameters can be for instance:
  - the I, B and respectively P frame rates.
  - MPEG/ATM encapsulation parameters : PCR aware/PCR unaware, MPEG2 to AAL5 transport stream packet adaptation issues, average per flow jitter.

  The active code based monitoring is performed on this type of objects. The content provider is able to check if SLAs parameters are satisfied by deploying application specific code. For instance, it can simulate client behaviour and measure the performance of the provided service.
- A DVPN member corresponds to a a network address, used by a particular residential user, and that is subscribed to DVPNs.

The dynamic reconfiguration of the tree is initiated at this level first and down-propagated to the network level. Several causes can trigger such a reconfiguration.

- users that join a DVPN,
- users leaving a DVPN,
- Management initiated actions. For instance, stationary parts of the multicast tree, can be merged onto a one-to-many permanent connection in order to facilitate management.

For instance, if one user desires to join a DVPN, its SAP asks the DVPN Tree manager to add a branch to the particular NFC. At the network level, this operation is translated in adding a branch to the trail supporting the NFC. Since a LayerTrail is build from interconnected SNC and LC, these objects must be created/modified. The usual operation flow involves the addition of branch to an existent SNC object that is already used to multicast the DVPN traffic, and the necessary LC and SNC objects are created in order to deliver it to the users's access point. In case of users leaving a DVPN, there might be the case that whole areas of the multicast tree need to be teared down in order to release bandwidth resources.

5 The distributed software architecture

The software architecture, illustrated in figure 3, is composed of two main parts. The first one regards the clients objects used to access specific interfaces on the service provider system, whereby the second one is concerned with the implementation specific issues in the service provider domain.

5.1 The Client side

The only object visible to the outside of the platform is the VPNMemberFactory object. Clients will use the Corba Naming Service to be able to reference this objet. Its main responsibilities are related to the authentification of users and the processing of their requests.

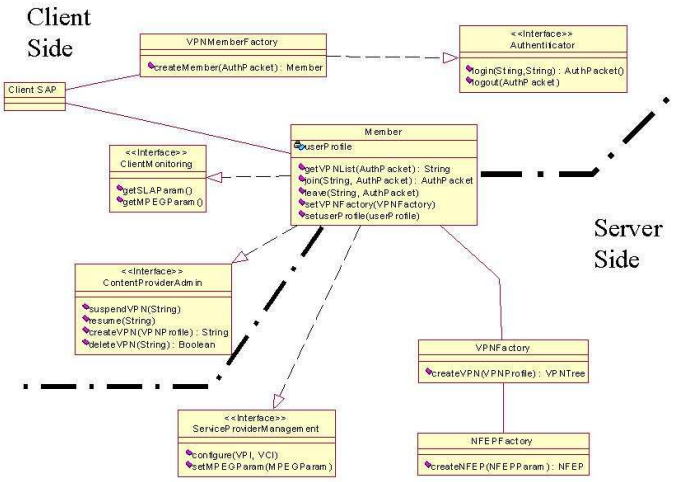

Fig. 3. Distributed Software Architecture

Authentication and service access interfaces The authentication part is dealt with by implementing the `Authentificator` interface which consists of two methods:

- `login(String, String)`. Using a centralized database where the client's profile is stored, and the provided userid and password, users are offered access to the service. In case of a successful login operation, an `AuthPacket` is returned to the user. This return value will be further used by the user for all his successive operations. In this phase of the project it represents the user's cresidentials in terms of subscribed VPNs, and specific Service Level Agreements.
- `logout(AuthPacket)` will simply logout the user and clean up resources.

The returned `AuthPacket` packet will be used to authenticate every operation. This is an additional security precaution since a malicious user could try a masquerade and obtain a reference to already instantiated `Member` objects to access and use the service at someone else's charges.

The main functionality of the `VPNMemberFactory` object is to create `Member` objects. This is done using the `createMember(AuthPacket auth)` method, which creates the `Member` object to be used by the user to join/leave particular VPNs. Additionally, users can use this object in order to

- get statistics concerning service usage and preliminary billing information.
- configurate simple policies in terms of VPN access. Access to a particular VPN could be prohibited for a delimited period of time based on the rating of

its information content. For instance, movies having a high degree of violence could be blocked in order to protect young watchers.

Operational and management interfaces The `Member` class implements one operational interface and three management interfaces. The operational interface is used to join and leave DVPNs.

- `getVPNList(AuthPacket)` which returns the list of VPN identifiers, that the user can subscribe to. This information will be used to configure some of the user's hardware.
- `join(String, AuthPacket)` connects the particular Member object used by a user having cresidentials given by the auth AuthPacket to the VPN identified by vpnId.
- `leave(String, AuthPacket)` leaves the current VPN, that the client is connected to.

The management interfaces provide for several types of management initiatives. The residential user, the content provider and the service provider are allowed to perform management actions. The first one, called `ClientMonitoring` permits the monitoring of SLA established parameters at the client side. Such type of parameters are for instance:

- parameters related to zapping. A simple SLA could specify that connecting to a particular VPN, as a result of a user zapping action, should not take more that 250 ms.
- parameters related to the MPEG stream like for instance the received I, B and P frame rates, and associated measurements.

A Content Provider is a special type of a user having more facilities in terms of rights to create and delete VPNs and perform restricted custom management on its VPNs. This is done through a private interface `ContentProviderAdmin` implemented by the `Member` class. A Content Provider can access this interface only if its `AuthPacket` packet encodes these type of cresidentials. The methods included in this interface permit to create/delete and manage a particular VPN by a content provider. Administrative issues (like client subscribing and unsubscribing operations) but also global MPEG/ATM related performance measures can be addressed, if necessary.

5.2 The Server Side

The `Member class` exports a management interface to the Service Provider to be used in the end user management and monitoring.

VPN end user configuration and monitoring The Service Provider can access the `ServiceProviderManagement` interface to perform management operations on to `Member` objects. This interface is accessed by the Service Provider

in order to perform application specific management configuration and monitoring. In the case of MPEG over AAL5 video delivery, such management pertains to the:

- configure hardware specific parameters (eg. interface,VPI/VCI).
- setting the value of MPEG transport stream packets, encoded in one AAL5-SDU. One transport stream packet is 188 bytes such that if only one packet stream is encoded in one AAL5-SDU, one needs 5 ATM cells for the their transport. If more transport stream packets are encoded in one larger AAL5-SDU, bandwidth is used more efficiently. The default value is 2. Several choices are possible based on jitter requirements. For instance, a large value will provide for efficient bandwidth utilisation requiring though a higher jitter.
- Setting the MPEG stream rate in conformance with the ATM Layer traffic description. Several mechanisms for this purpose are introduced in [1,11].

This interface is also used by the Service Provider in order to verify that SLA's in terms of residential users MPEG quality is assured. In order to allow for more flexibility for this purpose the Service Provider can deploy active code at the user's site, using this interface.

New DVPNs are created through a a `VPNFactory` object. The method `createVPN` is called on behalf of a `Member` object when a Content Provider deploys a new DVPN. The argument of this method is an object of the type `VPNProfile` encapsulating VPN related SLAs. In our case, they concern the MPEG-2 end-to-multi-end properties. These properties will be used to create the corresponding `NFC` object. The resulted `VPNTree` is registered with the `VPNForest` entity, and its identifier is returned to the user. A `VPNTree` object corresponds to the administrative view of a DVPNs. It includes methods to add/remove members, and provides for general membership related statistics (average time of connection, average number of users, max user number). Specific MPEG-2 related parameters, like the ones mentioned above, are accessed via the NFC object attached to a `VPNTree`.

Dynamic VPN membership management Let us consider the case of one client joining a DVPN, illustrated in figure 4 .

In this case the method add(client-member) is called on the corresponding `VPNTree` object. This results in the creation of a `NFEP` object, its addition to the associated `NFC` and its configuration according to the `NFC` parameters. Next, using the associated `Member` object, the `nwTTP` to be used is determined. A client profile includes a `nwTTP` determined by the access equipment and the triplet(port, VPI/VCI) used on the latter. The `LayerTrail` object used to support the `NFC` object is determined and required to add the `nwTTP` object. Since one can map a `LayerTrail` object onto a one-to-multipoint `SNC` objet, a similar Subnetwork Connection Management structure to the one introduced in [7] can be used. As soon as the `Member` object has been added to a DVPN, a packet of the type AuthPacket is returned. This packet will be used by the methods implemented

through the `ContentProviderManagement` interface, in order to authentificate management actions performed by the Content Provider. The global view of currently existent DVPN is available to the network manager via the `VPNForest` object.

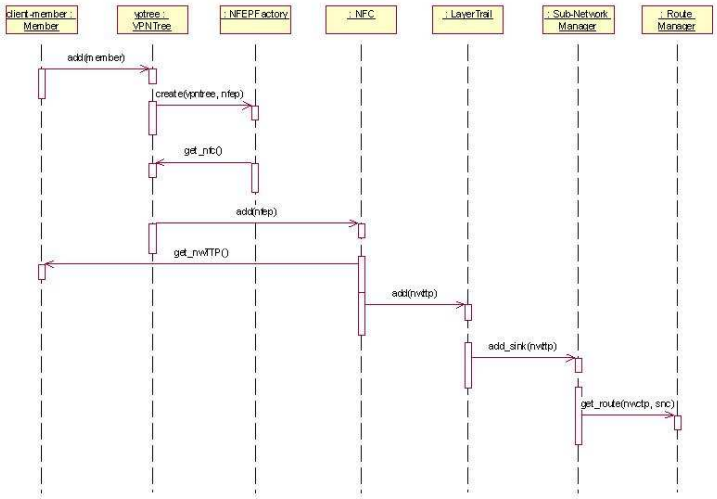

Fig. 4. DVPN join operations

6 Related work

From the architectural framework point of view, our approach relies on the APIs defined in the IEEE PIN project and at the Columbia University. Currently we base our specifications on the ATM Switch Resource Abstractions APIs. Concerning VPNs, the Genesis project [5] proposes a distributed network operating system based on those APIs as well as a spawning architecture for their setup. Their objectives are quite different from the ones defined in our framework. The Genesis kernel could be used in our framework to spawn an initial DVPN. The idea of combining programmable and active technology has already been proposed in the mobiware framework [6,4], where the programmable framework was used to build the signalling plane, and the active technology permitted to inject code to adapt the data plane to wireless customer-tailored specific QoS

conditions. The principle developed in our approach is conceptually similar but applied to the management plane for the active technology part. The use of active technology for management has been proposed several times over the last year. Most of these approaches deal with standard delegation, mostly in conjunction with a legacy SNMP environment. This is done for instance in the SmartPackets approach [9] where active packets are used to delegate monitoring and access to SNMP variables. Our approach for the active technology part is different from those approaches in the way that the active code sent to monitoring Execution Environments is not dependent on standard SNMP, but provides application specific flow monitoring facilities to instrument the edge probes.

7 Conclusion and future work

In this paper, we have presented a framework which combines both a programmable paradigm and the use of active technology for the monitoring of dynamic virtual private networks. The use of a programmable network architecture is motivated by the need to combine both application level signalling and network level management.

At the programmable network level we have implemented a DVPN model and a prototype configuration manager. This manager is fully integrated in the DPE, enabling thus efficient information exchange with the signalling facility.

Using active technology for dynamic monitoring represents in our framework an extension to management by delegation. In fact, using this technology enables deployment of management functions that are adapted to a given type of application, gathering the semantics of the data flows. This is very interesting for making service management more efficient and user-friendly.

The current approach is limited to multicast trees based on a single source for each VPN and multiple data sinks. One future direction of this work will be to extend the approach to multi-source multicast groups.

References

1. Audiovisual multimedia services: Video on demand specifications 1.1. *The ATM Forum*, March 1997.
2. C.M. Adam, A.A. Lazar, and M. Nandikesan. ATM Switch Resource Abstractions, March 1999. IEEE/WG P1520/TS/ATM-017 Working Document.
3. C.M. Adam, A.A. Lazar, and M. Nandikesan. Switch abstractions for designing open interfaces, March 1999. IEEE/WG P1520/TS/ATM-016 Working Document.
4. O. Angin, A.T. Campbell, M.E. Kounavis, and R.F. Liao. The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking. *IEEE Personal Communications Mag., Special Issue on Adapting to Network and Client Variability*, 5(4):32–44, August 1998.
5. A. Campbell, H.G. De Meer, M.E. Kounavis, K. Miki, J. Vicente, and D.A. Villela. The Genesis Kernel: A Virtual Network Operating System for Spawning Network Architectures. In *2nd Int'l Conf. on Open Architectures and Network Programming*, N.Y., May 1999.

6. A.T. Campbell, M.E. Kounavis, and R.F. Liao. Programmable Mobile Networks. *Computer Networks and ISDN Systems, Computer Networks*, 31, April 1999.
7. J. P Gaspoz. *Object Oriented Method and Architecture for Virtual Private Network Service Management*. PhD thesis, École Polytechnique Fédérale de Lausanne, 1996.
8. E.C. Kim, C.S. Hong, and J.G. Song. The multi-layer vpn management architecture. In *Proc. NOMS 1999*, pages 187–199, 1999.
9. B. Schwartz, W. Zhou, A.W. Jackson, W.T. Strayer, D. Rockwell, and C. Partridge. Smart Packets for Active Networks. In *Proc. OpenArch '99*, March 1999.
10. R. State, E. Nataf, and O. Festor. Poster session: A Java based Implementation of a Network Level Information Model for the ATM/Frame Relay Interconnection. In *Proc. NOMS 2000*, April 2000.
11. C. Tryfonas and A. Varma. MPEG-2 Transport over ATM Networks. *IEEE Communications Survey*, 2(4):24–32, 1999.
12. D.J. Wetherall, J.V. Guttag, and D.L. Tennenhouse. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols,. In *Proc. IEEE OpenArch'98*, 1998.