A Distributed Management Platform for Integrated Multicast Monitoring

H. Sallay, R. State, O. Festor
LORIA-INRIA Lorraine
615 rue du Jardin Botanique
54602 Villers-lès-Nancy
France
{Hassen.Sallay, Radu.State, Olivier.Festor}@loria.fr

Abstract

While multicast services are becoming very attractive, their large deployment and commercial use is currently slowed down partly due to the lack of integrated management solutions for the components that participate to the operation of these services at various levels. Excellent standalone components exist today and are good candidates for the integration. Joined and interfaced with standard management platforms, they cover most of the functions retaled to multicast service monitoring.

In this paper we present the resulting architecture of one integration effort which combines two multicast management tools for topology monitoring, pre-event testing and in-situ monitoring. The proposed architecture is used for service level monitoring and data collection.

Keywords

IP Multicast, Monitoring, MRM, JMX, Service Level Management

1. Introduction

Multicast services become more and more important within the Internet. Initially deployed within the MBone through a limited number of multicast routing nodes interconnected through tunnels, native multicast protocols are now widely deployed in most commercial routers and their use in various situations (content server update, video conferencing, collaborative work) is increasing continuously. The advent of new protocols such as PIM-SSM (Protocol Independent Multicast – Single Source Mode) [29,30], which instantiates a more simple and scalable model fitting many multicast usage scenarios, will undoubtly foster the wide acceptance and usage of these services.

While the interest in these services becomes large, the need to manage them becomes huge. If multicast has to be considered as one commercial service among others, integrated management solutions being able to control and monitor the

0-7803-7382-0/02/$17.00

service and its components to guarantee negociated service levels need to be provided. Our paper addresses this issue of managing multicast communication by proposing an integrated management architecture for this purpose. Our paper is structured as follows: we present in the section 2 an introduction to current approaches for the management of multicast enabled networks, including a short overview of the multicast reachability protocol which provides the general framework used in our work. Our approach for a toolkit integration dedicated to the management of multicast networks is described in section 3. Section 4 describes the implementation of our approach. Finally, section 5 concludes the presented work and outlines to future developments.

2. Multicast Reachability Monitoring

Until recently, the management of multicast services relied on a set of diverse often standalone tools. Some of them like Mhealth [9], rely on the Real Time Protocol (RTP [22]) to collect multicast data. In fact, these monitoring solutions use RTCP (Real Time Control Protocol) [22], the control protocol accompanying RTP, in order to gather membership information of one or several groups. Msessmon, Mlisten, and RTPMon [5] are typical examples of this family of approaches.

Mtrace [17] is the multicast version of the traceroute utility. It can be used to infer the multicast tree topology. The Mtrace utility works by tracing the path from individual receivers upstream towards the source. Routers located on the path from the receiver to the source are supposed to answer a particular type of requests. While Mtrace is very useful, it has several limits. For example, it has been shown that such an approach might not be adequate when the same session is monitored by several Mhealth tools [7]. A second drawback of this approach is the high workload induced on the routers located close to the source since they have to reply to requests concerning almost all receivers.

A distributed fault management scheme for multicast communication has been proposed in [7,8]. There, software entities called session watchers perform monitoring operations for the path linking the watchers to the source. Monitoring optimisation is done by ensuring that path segments common to several watchers, are monitored only once. This efficient approach lowers the workload on the routers and monitoring related traffic on the network.

Some other approaches are more focused on configuration management. Mrinfo [5] can determine the set of tunnels and their status for a particular router. Using SNMP in the context of managing multicast can be done by several tools. Mstat [5] allows to query an individual router, whilst Mview [5] can visualise the Mbone topology and provide some performance information. Another tool is MMON (Multicast Management Tool) which is part of the HP Network Node Manager. It uses both Mrinfo and SNMP to discover the multicast topology of a domain.

Several approaches consider the use of application level information for multicast management. Mantra [15] is such a tool that collects and summarizes multicast information regarding large multicast groups. This is done using remote site scripts

invoked to populate tables in the server concerning active sessions, membership, group statistics as well as multicast routing information.

The above mentioned approaches assumed that access (SNMP, or IGMP) to the network equipment is possible. A sophisticated method based on the Maximum Likelihood Estimator has been developed to be used in a context where direct access is impossible [35]. Such a scenario can be encountered when multicast traffic is routed through a private transport network. The starting assumptions were that only endpoint measurement (packet loss and variation delay) is available. The three problems addressed the issue of estimating the underlying topology [10, 13], the loss probabilities on individual links [11] and the packet delays for particular links [12]. This type of work is now part of a larger AT&T project named Multicast Inference of Network Characteristics (MINC) [16] aiming at a broader application of inserting traffic probes to estimate network conditions.

A promising interesting approach for building an integrated management platform could be based on the MRM (Multicast Reachability Monitoring) and its associated architecture and protocol [1,4]. This management architecture is based on the cooperation of three major entities. The first one is a manager responsible to execute tests on a network. These tests are used to detect faults and verify traffic assumptions between a source and several receivers. This manager is the interface that users (network managers) must invoke in order to perform the management actions. The source and the receivers used to test multicast traffic are represented by entities called Test Receiver (TR) and Test Sender (TS). A Test Sender will send multicast traffic according to requests issued by the manager. These requests are called Test Sender Request (TSR). This traffic is received by the Test Receivers. The latter are instructed by the manager how to collect and report management information about the received traffic. These instructions are provided by Test Receiver Requests (TRRs).

A Test Sender request specifies how a TS should generate test packets. The TRR specifies how a TR should collect the reception data. The MRM manager periodically transmits beacon messages to advertise its liveness to all MRM testers. MRM is an extremely useful tool to monitor multicast routing operation and to assist in troubleshooting anomalies and connectivity problems. It is designed to provide a set of functions complementary to what can be obtained by other commonly used tools. We chose this architecture as the middleware for dynamic monitoring deployment.

3. Management Architecture

The functional architecture of the platform is made of three main functions based on three components: a topology service based on a topology manager, a monitoring service based on an MRM management model and a test manager and finally a graphical user interface providing an integrated management presentation function (see figure 1). The platform allows the definition of test operations to be made online by a network operator. This definition includes a set of general templates that are specialised for individual customised tests and the individual tests with their

associated results. These tests are in fact instances of MRM test sessions.
The presentation function takes the form of an interface to the human operator. This interface offers an integrated view on the multicast configuration and the underlying network topology. The multicast management information is defined as a set of static information mostly related to multicast routing configuration and of dynamic information associated to generated multicast traffic. The static information is provided by the topology service. The dynamic information is given by an entity called MrmDomain Manager. The term domain should be read with the following in mind. We consider the task of managing one administrative domain, ignoring for the issue of multi-domain management in a first step. For efficiency this domain can be split in several parts, called MRM sub-domains. Each MRM sub-domain is locally managed by a sub-domain Manager.

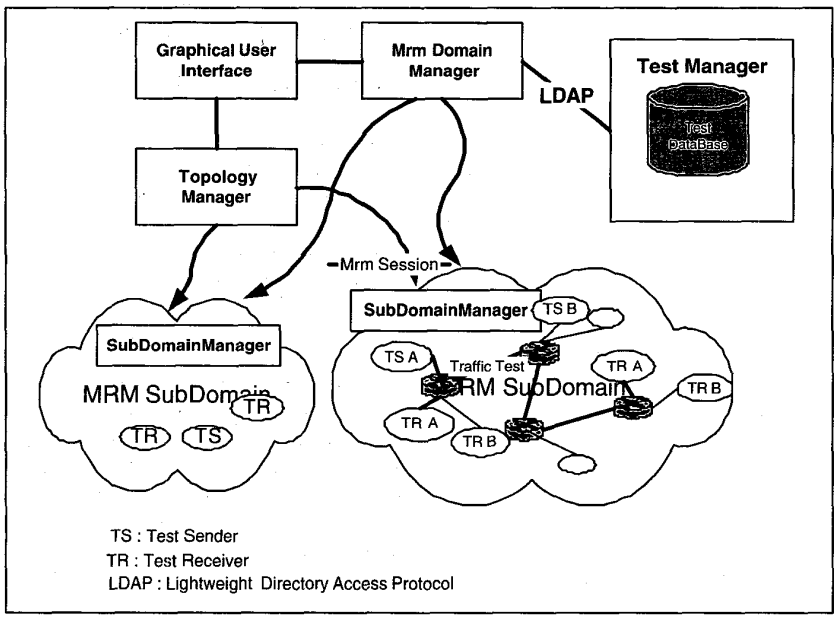

Figure 1: Management Architecture

All entities that realise the different functions of the platform can be distributed across the network.

3.1 The MRM Management Model

Figure 2 shows the management information model that defines the interface to MRM setup and operation. The *MRMDomainManager* class provides the entry-point to the management API (Application Programming Interface) responsible to configure a set of multicast receivers and sources. A test is represented by the *Session* class. All the *Sessions* are included in the *MRMDomainManager* class. The latter exposes an API allowing to add/delete sessions, start or stop an individual session.

For instance:

- *addDomainSessions() /deleteSession()* creates/adds or deletes a session in one or several Mrm domains. Each session will define a configuration for the Mrm test session. A configuration specifies the test senders, the test receivers and all the parameters of the scenario of the test.
- *TerminateTest()* will terminate a test.

The *Session* class is characterised by some attributes like a state (activated, suspended, terminated) that can be modified and consulted. Its API allows to:

1. Add or remove a sender/receiver,
2. List current senders/receivers,
3. Configure the traffic descriptor for the multicast test,
4. Start/Stop a test.

From an information modeling perspective this class is particular important because it represents several *SubDomainManager* classes jointly managing a multicast communication. Thus, a multicast session that spans several network regions, each of them under the control of a *SubDomainManager*, can be represented in the management model. Each *Session* is associated (indirectly via the *SubDomainManager* class) to a list of participating *TestProcesses* (ie. MRM Agents) in this sub-domain. The latter can be either used to simulate a sender or a receiver. For each *Session* there is one IP multicast address defining the group multicast that it covers. The configuration of one session concerns the specification of a list of *SubDomainMangers*, *TestProcesses*, their roles, the IP address of the Multicast group, the duration of the session, the start-up time of the test and the frequency of getting the management reports. For instance:

- *SetTestSenders()* and *GetTestReceivers()* will configure and respectively collect the test senders and receivers status.
- *setTsrs()/getTsrs()* and *setTrrs()/getTrrs()* are used to communicate between the test senders and test receivers. To collect and analyze the information, the *GetFaultStatistics()* and *displayReport()* operations are used.

The *SubDomainManager* class represents the management client in charge of a dedicated network region. It provides an interface towards individual applications acting as TestSenders and TestReceivers. Each *SubDomainManager* has a complete view of the sessions that are initiated only in its sub-domain and a partial

view of the sessions that are defined on several sub-domains. While the *SubDomainManager* provides the list of all the *TestProcesses* belonging to the different sessions actived in its sub-domain, using the *IPAddressMulticast,* the *Session* can get all the active *TestProcesses* belonging a session but they are located in different sub-domains.

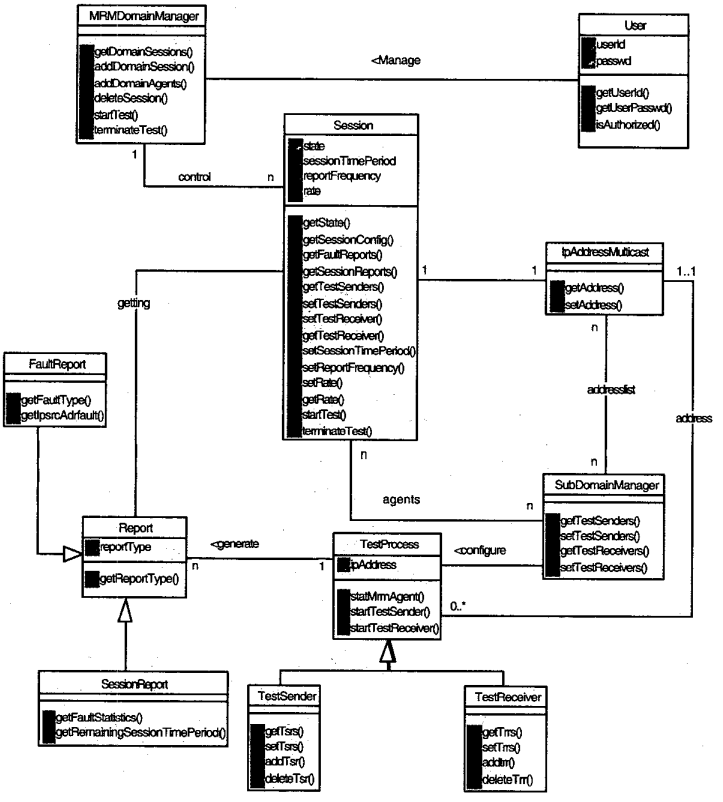

Figure 2: Management Information Model

The operation flow to start a test will be as follows:

1. The operation *startTest(Asession)* is invoked on a *MRMDomain Manager* instance,
2. For the *Session* object Asession, all sub-domain Manager are retrieved. This is done to specify the network regions that will cover by the session.
3. For each *SubDomainManager*, the list of associated *TestSenders* and

TestReceivers is received.

4. The *startMRMAgent* operation is called on each object obtained in the previous mentioned list.

Reports are modelled by the *Report* class. Two types of specialisation are defined for this class. The *Sessionreport* contains the state of the session, its evolution over time and the percentage of outages. The *Faultreport* contains the information for the fault notifications.

3.2 The Topology Manager

The global view of the underlying network and multicast tree is maintained by this component. For this purpose, we use a topology oriented network information model. which represents a logical view of the network resources used by the service. Several network information models, have been already defined (see [23], [24] [25]). Based on these proposals, we chose a simplified representation as shown in figure 3.

We decided to keep the model as simple as possible in order to allow its usage within a dynamic environment, providing enough power to model a multicast tree.

At a higher level of abstraction, a multicast tree can be seen as a *LayerTrail.* The members of this tree are modeled by the *nwTTP (*network Trail Termination Point*)* class. Parameters of the *LayerTrail* class are related to the particular multicast group, like for instance the IP multicast address. The *nwTTP* is mostly a holder for the IP address of individual members.

At a next level of abstraction we can decompose the multicast tree on the underlying network topology. At this level we retrieve three major classes.

The *LC* class represents a particular multicast tree routed between two adjacent routers. It is delimited by two *nwCTP* (network connection termination point). The *nwCTP* class is used to represent the network address used by a router forwarding multicast packets on a particular interface.

The *SNC* (subnetwork connection class) class represents the forwarding done in a router for one multicast tree. It allows to identify the interfaces and the next hop routers to which an incoming packet is forwarded. Even though such a class is more useful in a connection oriented type of network, we keep it in the model for simplicity reasons. It allows the to reuse of information models that we have worked with [6], without major changes.

At the third level of abstraction we have the physical network. At this level, we model routers and links between them.

A router is represented by the *Subnetwork* class. Physical links are modeled by the *Link* class. Typical attributes for the *Subnetwork* class are hostname and location. Attributes for the *Link* class, are related to the capacity of the physical link.

Interfaces delimiting a *Link* object, are modeled by the *LinkTP* (*Link Termination Point*) class.

One important feature of the proposed network information model is the usage of stereotypes. Without detailing to much, stereotypes are used in order to express the constraints that are inherent in multicast enabled communication. For instance, a

LayerTrail is decomposed into *SNC* and *LC*. However, this decomposition has to make sure that cycle-free and fully connectivity properties are valid for the instantiated models. These properties are expressed in terms of object model navigation over associations.

We capture these constrains via OCL (Object Constrained Language) [26], the official (OMG proposed) companion of the UML [27] .

A set of constraints concerning several classes are associated to a stereotype.

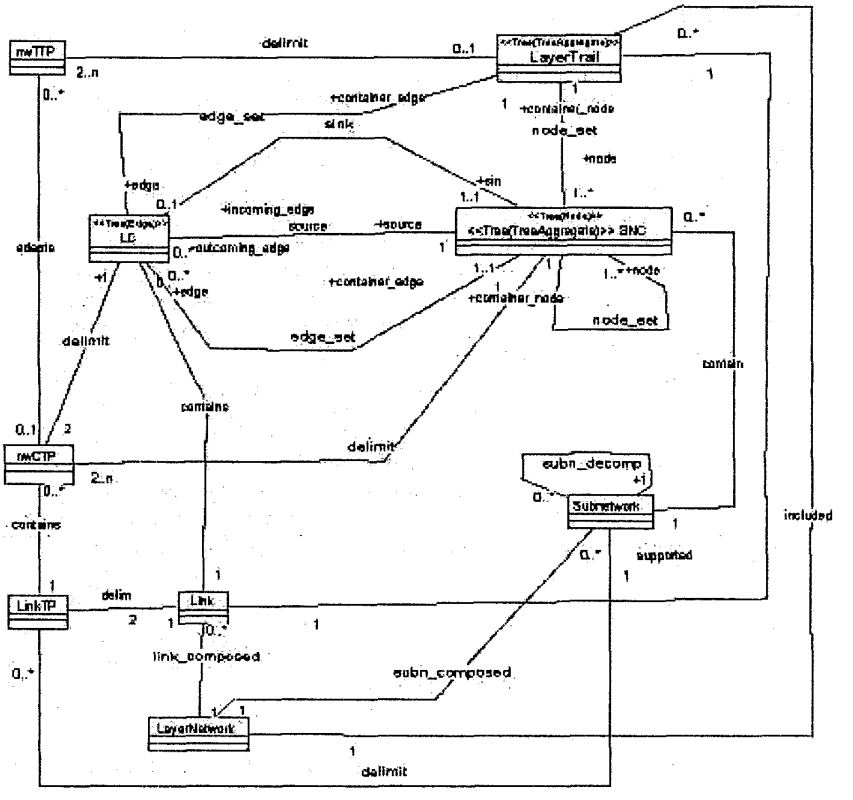

Figure 3: Network Level Information Model

These stereotypes enable the specification of the individual role played by a class with respect to some decomposition. For instance, a *LayerTrail* is the entity which is decomposed (a tree), whilst its nodes are given by *SNC* type of objects.

The edges in the tree are represented by *LC* objects. Associations relating edges to nodes (incoming, outcoming) are added for a complete and robust modeling. This OCL enhanced modeling allows to automate the use of tools that verify at run time that all constraints are satisfied.

3.3 Test Manager

The TestManager component is used to configure the TR and TS elements for a test session. It is based on a LDAP directory [20] used as a database for the configuration information. The benefits of using LDAP directory services for the storage of data for the network are already motivated by the DEN approach [21].

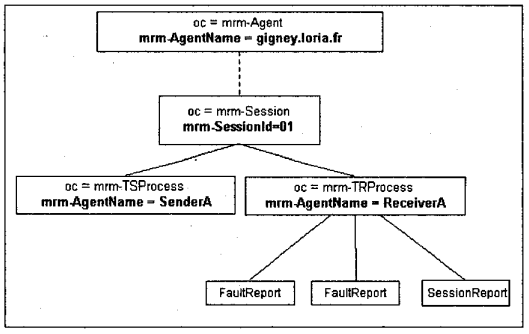

Figure 4: LDAP information tree for test process information storage.

For our purpose, we have mapped the configuration features of our information model to a LDAP schema enabling the directory to store a test session configuration data. In addition to the configuration information, we have also mapped fault and session reports classes, in order to store in the directory the reports issued from MRM agents.

Based on our LDAP schema, each MRM test schema is stored in the Directory Information Tree (DIT) as shown in figure 4. Session configuration information are then retrieved from the DIT by each MRM Domain Manager. This is done through a Java API (called MrmJavaApi) we have developed on top of the JNDI API (Java Naming and Directory Interface). The DIT is also accessed by each TR process to store fault reports. The main classes of our API are:

- *MRM_ManagerConnector*: it is used by MRM domain manager implementation to connect to a LDAP server and then to retrieve or store a session configuration;
- *MRM_Session*: it gives a Java representation of a session as a list of MRM_TRProcess instances and a list of MRM_TSProcess instances;
- *MRM_AgentConnector*: it is used by MRM agent implementation to

connect to a LDAP server and then to store a MRM report;

- *MRM_Report* together with its its subclasses MRM_FaultReport and MRM_SessionReport: give a Java reprensentation of an MRM report.

4. Implementation

We use two tools for instantiating and updating the model. The first one is the Mrinfo [5] program that allows to retrieve the multicast configuration of one particular router (obviously, this can be done only if that router responds to the *ASK_NEIGHBORS* igmp message).The second tool is Mtrace [5]. This program allows to trace a route from a receiver to a source working backwards using a particular multicast address.
The MRM agent builds and updates the network information model by the invocation of these two utilities. This invocation is done using native code and is the only place in the software architecture not being pure Java.

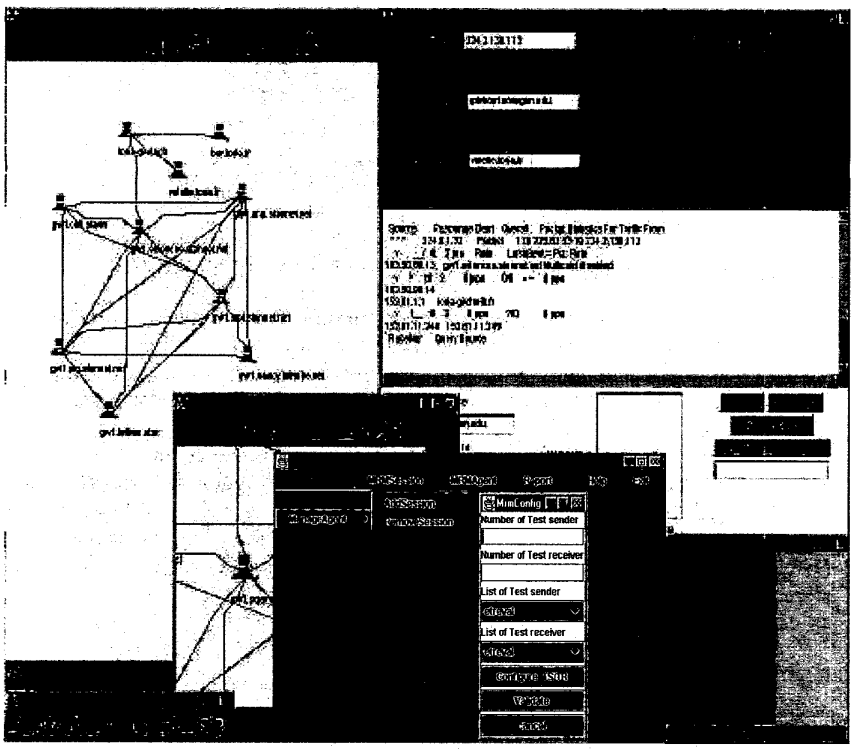

Figure 5: the manager side GUI

All service components (MRM Agent, MRM Manager, topology manager) have been developped using the JMX (Java Management eXtension) Framework [36]. All objects have been defined as MBeans and the services communicate with the Management application using Java RMI (Remote Method Invocation).
A screenshot from the implementation is shown in figure 5. The GUI is a Java application developped with the Koala Graphics toolkit [14] to represent the topology and ongoing communication within a MRM domain.
Three main areas are of interest in this schema. Region 1 (upper left) shows the static multicast information obtained by checking multicast routing tables. The display of this information is done through several views, covering different areas of the map at different zoom scales. This enables to zoom in, where network topology is dense, or where detail is needed, without having to do it for the whole map. Figure 5 shows the network topology located in the immediate vicinity of our campus network.
Region 2 (upper right of the screenshot) is a simple GUI interface to mtrace.
The component used to configure a MRM/MRM agent is shown in the third region (lower center). The manager uses this interface to invoke management actions and retrieve some of the network related information.

5. Conclusion

Multicast services are among the most important core services for future applications. Being able to manage these services at both network and service level is one of the challenges that the management community has to face. To build an integrated management architecture that addresses these issues, we have proposed the integration of the Management Reachability Monitor in a larger framework for multicast service management. To this end, we propose a management domain model for multi-domain MRM configuration and data collection together with a distributed environment for managing these domains. The test and service monitoring facilities have been integrated with our multicast topology monitoring engine to allow easy configuration of network nodes and MRM daemons. All components have been implemented in Java using the JMX framework and are used to monitor multicast activities within our lab and beyond the campus network.
The proposed framework represents our first integration level for multicast monitoring. We are currently investigating the integration of additional components in the environment. Of specific interest are: the evolution of our network model towards the CIM (Common Information Model) network and service models; the extension of MRM to support user specific evaluation capabilities, i.e. algorithms which evaluate a user's satisfaction rather than a UDP level packet count. We already proposed an algorithm for the lattest in the case of multicast video transmission in [34]. Also the transition to IPv6 and influence of protocols such as MLD [32] or REUNITE [33] on the management of multicast sessions is of particular interest to our research.

Acknowledgement

We would like to thank Mr. Nizar Ben Youssef for helpful discussions and suggestions concerning the directory service integration for the prototype.

References

[1] D. Farinacci, K. Almeroth, L. Wei. Internet Draft: Multicast Reachability Monitor MRM, October 1999.

[2] L. Wei, K. Almeroth. Internet Draft: Justification for and use of the Multicast Routing Monitor (MRM) Protocol, August 1999.

[3] Sun microsystems. White paper: Java Management Extentions: Dynamic management for the service age, June 1999.

[4] K. Sarac, K. Almeroth. End host implementation of MRM.

[5] K. Sarac. and K. Almeroth. Supporting Multicast Deployment Efforts: A Survey of Tools for Multicast Monitoring., Journal of High Speed Networking--Special Issue on Management of Multimedia Networking. 2001.

[6] R. State, O. Festor and Nataf. Managing concurrent QoS assured Multicast sessions using a Programmable Network Architecture, 11th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2000), Austin TX, December 2000, Springer LNCS

[7] Anoop Reddy, Deborah Estrin, Ramesh Govindan. Fault Management in Multicast Trees, ACM SIGCOMM, Stockholm, Sweden, August 2000.

[8] Anoop Reddy, Deborah Estrin, Ramesh Govindan. Large Scale Fault Isolation. IEEE Journal on Selected Areas in Communications, special issue on Network Management, vol. 18, no. 5, 2000.

[9] Makosfske, K. Almeroth. MHealth. A Real time Multicast tree visualization and monitoring tool. Proc. NOSSDAV'99.

[10] R.Caceres, N.G. Duffield, J. Horowitz, F. Lo Presti, D. Towsley. Loss-based Inference of Multicast Network Topology. CDC'99

[11] R.Caceres, N.G. Duffield, J. Horowitz, D. Towsley. Multicast-Based Inference of Network Internal Loss Characteristics. IEEE Transactions on Information Theory.

[12] N.G. Duffield, F. Lo. Presti. Multicast Inference of packet delay variation at interior network links. In Proc. IEEE INFOCOM 2000.

[13] S. Ratnasamy, S. McCanne . Inference of Multicast Routing Trees and Bottleneck Bandwidth using End-to-end Measurements. In Proc. IEEE INFOCOM 1999.

[14] Koala Graphics. www.dyade.fr.

[15] P. Rajvaidya, Almeroth, K. "A Router-Based Technique for Monitoring the Next-Generation of Internet Multicast Protocols", International Conference on Parallel Processing, 2001.

[16] A. Adams, T. Bu, R. Cáceres, N.G. Duffield, T. Friedman, J. Horowitz, F. Lo Presti, S.B. Moon, V. Paxson, D. Towsley, The Use of End-to-End Multicast Measurements for Characterizing Internal Network Behavior, IEEE

Communications Magazine, to appear

[17] Fenner, W., Casner S. « A traceroute facility for IP Multicast, <draft-ietf-idmr-traceroute-ipm-07.txt>, July 2001.

[18] J. Walz, Multicast Monitoring – Current Usage and a New Hierarchical Protocol –, Master Thesis, Dept. of Computer Science, University of Massachusetts, February 2001.

[19] Laurent Andrey, Olivier Festor, Emmanuel Nataf, Radu State. JTMN: A Java-based TMN Development and Experimentation Environment IEEE Journal on Selected Areas in Communications,Vol 18., No. 5., May 2000

[20] T. Howes, M. Smith and G.S. Good. Understanding and deploying LDAP directory services. New Riders 2001.

[21] J. Strassner. Directory Enabled Networks. Macmillan Technical Publishing, 1999.

[22] RTP/RTCP: http://www.networksorcery.com/enp/default0304.htm

[23] C.-C. Shen and J. Y. Wei, "Network-Level Information Models for Integrated ATM/SONET/WDM Management," IEEE/IFIP 1998 Network Operations and Management Symposium, New Orleans, USA, 1998.

[24] Network Resource Information Model. TINA-C, 1997.

[25] ITU-T, Generic Network Information Model. Recommendation M-3100: ITU-T, 1992.

[26] J. Warmer and A. Kleppe. The Object Constraint Language: Precise Modeling with UML, Addison-Wesley, 1998.

[28] J. Rumbaugh, I. Jacobson and G. Booch. The Unified Modeling Language Reference *Manual*, Addison-Wesley, 1998.

[29] H. Holbrook, B. Cain Source-specific Multicast for IP, Internet Draft, November 2000.

[30] S. Bhattachayya, C. Diot, L. Giuliano, R. Rockell, J. Meylor, D. Meyer, G. Shepherd, B. Haberman, An Overview of Source-Specific Multicast (SSM) deployment, Internet Draft, May 2001.

[31] Rajvaidya, P. and Almeroth, K. and Claffy, K. A Scalable Architecture for Monitoring and Visualizing Multicast Statistics, Proc. IFIP/IEEE DSOM'2001, Ambler, A. and Calo, S. and Kar, G. (Eds)., LNCS 1960, pp.1-12, 2000.

[32] R. Vida L. Costa, R. Zara, S. Fdida, S. Deering, B. Fenner, I. Kouvelas, B. Haberman "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", <draf-vida-mld-v2-01.txt>, july 2001.

[33] I. Stoica, et al., "REUNITE: A recursive Unicast Approach to Multicast", Proc. Infocom 2000.

[34] R. State and O. Festor Active Network based management for QoS assured multicast delivered media, Proc . IEEE Joint 4 th. International conference on ATM and High Speed Internet (ICATM2001).

[35] N. G. Duffield and M. Grossglauser: Trajectory Sampling for Direct Traffic Observation

[36] JMX homepage: http://java.sun.com/products/JavaManagement/