A Framework for Management and Control of Distributed Applications using Agents and IP-multicast

Peter Parnes, Kåre Synnes, Dick Schefström
Luleå University of Technology/Centre for Distance-spanning Technology
Department of Computer Science
971 87 Luleå, Sweden.

Abstract— **As more and more applications on the Internet become network aware the need and possibility to remotely control them becomes larger. This paper presents a framework for control and management of distributed applications and components. This is done using IP-multicast and an agent based application architecture. The target of the framework is to allow for resource discovery of both controllable elements and available control points in the these elements as well as real-time control. All this is done in scalable and secure way based on IP-multicast and asymmetric cryptography. The presented framework is also independent of the underlying transport mechanism to allow for flexibility and easy deployment. The framework bandwidth usage and introduced control delay is presented. Details on the reference implementation of the framework and example usage scenarios where the framework is used to create bandwidth adaptive applications and better group awareness is also presented.**

Keywords— **IP-multicast, distributed management, control, secure messaging, reliable multicast, distributed applications, intelligent agents, quality of service management, Java**

I. Introduction

With the current increase in the number of deployed distributed applications the need for control and management grows. A central issue in any computing environment, both in academia and industry, is how to control and manage running applications. This is especially applicable and important to the more and more used application family of IP-multicast [1], [2] based distributed real-time applications for primarily desktop conferencing, distance education and media broadcasts with many simultaneous users.

These distributed applications usually utilize the available bandwidth more than traditional single user applications and they are usually more sensitive to large delay and jitter in the network. It is therefore very important that real-time media applications do not compete with each other for the available bandwidth but instead co-operate and try to utilize the bandwidth in the best possible manner.

When users start using real-time distributed applications, the risk for users doing the "wrong thing" and flooding the network with to much data also increases. Historically, this have been handled by locating the application, host and responsible user that generates the extra network traffic and asking the user to either terminate the application or lower its network usage (or in a UNIX environment, the system administrator may even just log into the host and terminate the application in question). The issue of finding the responsible user becomes more complicated when distributed applications are used in confederation between several organizations and over the Internet.

This work was supported by the Centre for Distance-spanning Technology (CDT), Luleå, Sweden

We have found that there is a need for a control *framework* where applications can be remotely controlled. The use of bandwidth might be one of the most important issues as that usually affects many users. Other important control scenarios include remote user support where the support people can get information about how an application is configured and can remotely change this configuration.

This paper presents and discuss a general framework for management and control of distributed applications. The target is that the framework should not end up in one isolated implementation but instead several different inter-operable implementations should emerge over time.

The rest of this section presents an overview of the distributed mStar environment from where much of the work presented in this paper have come, current problems and related work. Sect. II presents the new control and management framework. Sect. III presents how the framework can be and is currently used. Sect. IV presents the reference implementation of the framework including the mManager application and some future work. The paper concludes with a summary and discussion in Sect. V.

A. The mStar Environment

Since 1995 we (CDT) have been developing the *mStar environment* [3] which is an environment for scalable and effective distribution of real-time media and data between a large number of users. mStar is a shared environment that can be used for a number of different distance-spanning activities such as net-based learning (distance-education) and distributed collaborative team-work. It includes support for a number of different media including audio, video, shared whiteboard, distributed presentations using the World Wide Web and much more. It also supports on-demand recording and playback of sessions using either unicast or IP-multicast. The idea and need for a management framework came from the daily usage of the mStar environment at CDT. (Note, that the mStar environment is now being commercialized and sold under the name MarratechPro by the company Marratech AB in Sweden.)

B. Current Problems and Framework Requirements

When a distributed desktop conferencing application, such as the mStar environment, is deployed in a large organization; a

number of new management and control issues evolve. The administrators need to be able to get information about and control: which users are part of which conferencing sessions, which media in each session do they currently have active (i.e., which media are they currently receiving), if the user is currently transmitting any data within a session and then with which settings (e.g., for video transmission the settings in question would be the bandwidth currently used, frames per second or codec/video-format). If this information is available it will allow the administrator to control both session membership (e.g., kick out unauthorized members) and the total bandwidth used by this group of applications. This means that the network administrators can control the total amount of bandwidth used by each user and session explicitly.

To be able to monitor and control running applications, it is necessary to get feedback on what applications users are currently running, which version of the specific applications they are running, and the current configuration of these applications. Secondarily, it is necessary to be able to remotely control the applications. These problems can be divided into two major groups, *information reports* from users and *remote control* of applications. The identified problems lead to a number of requirements:

- The framework have to support large groups of applications, both for reporting and control.
- The framework have to support efficient control of groups of applications without the need to send control messages to each application explicitly.
- The framework have to protect users from unauthorized control of their applications.
- The framework have to allow developers to easily insert control access points in their software.
- The framework should be as independent of the underlying software and transport technology as possible.
- The framework should allow for scalable resource discovery of both available control objects and controllable variables and methods in these objects.

This paper focuses on presenting a new framework for addressing these problems and requirements.

C. Background Information and Related Work

This section presents related background information and some selected related work.

C.1 IP-Multicast Applications

Traditionally, distribution of multimedia data on the Internet to a group of users have been done using unicast, either by each multimedia application sending one exact copy to every receiver or by pushing the problem into the network and using a so called *reflector* which duplicates streams to every registered receiver. This means that duplicate data will be sent if the path between the sender and its receivers share common links in the network.

The power of IP-multicast is that data is only copied in the network where needed when sending the same data to several different receiving hosts. IP-multicast traffic is sent using UDP which is best-effort and in turn means that packets can be lost in the network. This might be a problem in control situations as the manager wants to be assured that sent control messages actually are delivered to the destination. This problem is further discussed in [4]. A drawback with IP-multicast is that it requires support from routers in the network to handle this special kind of traffic. If the routers in question are fairly new, it might be simply a question of turning on the support in the router software. To summarize, the IP-multicast solution saves large amount of bandwidth in the network.

C.2 Simple Network Control Protocol - SNMP

The *Simple Network Control Protocol - SNMP* [5], [6] is designed for control of network elements and basic applications. It is designed with a 'polling' architecture in mind meaning that the managing software have to request information from each element to be monitored. This architecture allows managers to get information from a monitored object and set variables in that object.

SNMP includes support for so called trap's where a manager can request to be notified when some predefined situation occurs. A restriction with these trap's are that they cannot be defined dynamically but the manager has to rely on predefined trap's. Note, that although this trap mechanism exists the normally used part of the SNMP mechanism is still only the "get/set" functionality.

As SNMP is designed to control a single element at a time, it is not really suitable for controlling large groups of real-time applications. Another important aspect not supported by SNMP is resource discovery. Further, every user that want to control and fetch information from a device have to have a corresponding definition document called a management information base to know what variables are actually available in the device to be controlled.

C.3 The Service Management System - SMS

The designers of *the Service Management System - SMS* [7] argue that the poll-architecture of SNMP causes managers to often notice problems too late as the object in trouble cannot notify its manager about its condition. The creators of the SMS system try to approach this problem by creating an architecture where each managed object is encapsulated by a SMS wrapper which marshals commands to and from the managed object. The wrapper has a SMS agent to aid it with automatic handling of certain situations. This SMS agent can be controlled by a set of rules (defined using the so called *Service Management Agent Programming language - SMAP*) to react on information provided by the SMS wrapper.

Although this architecture seems promising it does not include support for resource discovery, information reporting and scalable control of large groups of applications.

C.4 The Conference Control Channel Protocol - CCCP

Reference [8] presents the *Conference Control Channel Protocol - CCCP* which is a protocol for control of distributed multimedia applications. It consists of a text based message protocol for primarily control. The CCCP is similar to the work presented in this paper but differs in that we focus on a wider range of applications than just the group of conferencing applications and not only real-time control but also scalable reporting of information.

Reference [9] presents a similar message platform but that one is even more constrained and only target the problem of how to communicate between similar applications within a single host.

C.5 Java Specific Platforms for Distributed Applications

There exits a number of proposals for Java based architectures for distributed applications such as the InfoBus [10], JavaSpaces [11] and iBUS [12]. Some are very promising and flexible but they all make one large assumption on their programmatic environment and that is that they are very tightly integrated with the Java programming language and its runtime environment, They all assume that they have access to features that are very specific to the Java environment, such as Java-events and/or serialization (binary representation of Java runtime objects). This assumption makes these frameworks virtually unusable in other environments.

C.6 Other Potential Control and Management Technologies

Several other technologies could be used as the underlying mechanism (such as Corba [13], MPEG-4 DMIF [14], and SS7 [15]) but during our investigations we have found that all of these are either not scalable enough (centralized solutions where the central point becomes a bottle-neck) or to specific to their original design domain.

II. The Control and Management Framework

The purpose of proposing a new framework is to support developers in the process of creating a new kind of applications that are network aware and fit better into the global Internet. This framework includes a set of building blocks that allows for scalability in resource discovery, information reports and real-time control.

As presented earlier there are a number of problems involved in how to control and manage real-time distributed applications. This section presents a new control and management framework for addressing these problems and requirements.

A. Information Reports

To be able to request information from currently running applications there must exist a platform that allows for reports to be sent in a scalable way. With scalable, we mean that the solution found should be usable within sessions with a few users and within sessions with a large number of users, as well as sessions that run locally and over wide area networks. The amount of bandwidth needed in these different situations should of course be kept as low as possible. If a large number of applications send reports at the same time, momentarily a larger amount of bandwidth will be used as the total number of messages increases linearly with the number of users. The obvious solution is that every application should not send its report at the same time as every other application, but instead utilize a back-off and delay method based on the available bandwidth, the current number of members in the session and other reports received.

Reference [16] presents a mechanism for calculating the delay between control messages based on the mean size of the messages, the available bandwidth and how much of that bandwidth is reserved for control messages. The result of this mechanism is that the total amount of bandwidth used stays approximately constant independently of the size of the messages and the number of users in the session. This mechanism is reused in the manager framework (see Sect. II-E for more details about the delay calculation) with the difference that a larger portion of the bandwidth is allocated for the control and information messages to allow for faster interaction. Using this dynamically calculated delay, information reports can be sent to the whole group in a scalable way.

This reporting system can both be used for *regular reports* where applications report predefined information in regular intervals and for *on-demand reports* where applications report information based on requests. These two cases can also be combined where a manager requests applications to include some information that changes often in each regular report sent. This is for instance used for retrieving the amount of bandwidth currently being used by each member within their session.

To allow for administrator mobility all reports are always sent to the whole group using IP-multicast. This allows for different manager applications to be active within the same session at the same time without requiring that information is duplicated in the network (by different applications requesting the same information). This means that administrators can monitor and control their system from any host within the network.

B. Control

A pre-requirement is that the system should be able to control both single applications and groups of applications. Of course, the latter includes that if the same parameter is to be sent to all the applications only one message should be sent and not one specific and identical message to each single instance. This leads to that messages should be sent using IP-multicast within the group just as the earlier discussed information reports.

To allow for easy messaging, a message and control protocol is needed. This control protocol have to support dynamic addressing of applications and parts of applications. Here we propose an agent based architecture where developers can easily reuse components and agents within different applications. An agent is a software component that resides within an application and is responsible for one specific task. For instance, a video-agent would be responsible for capturing and displaying video data and a database-agent would act as an interface to a database-engine. Numerous agents can and are normally deployed within one and the same application. Fig. 1 shows some example agent based applications (especially note how the same agents are being reused in several different applications).

To support for dynamic addressing of applications and agents the *Control Bus - CB* [17] was chosen. The CB is designed for transmitting messages both within applications and between different instances of applications. The control bus is not tied into any specific transport mechanism but it is designed to be used together with any underlying reliable transport protocol.

The format of the CB messages are based on the message formats presented in [8] and [9]. Each message is completely text based and a message consists of four sections:
1. The *from section* which identifies the sender uniquely.

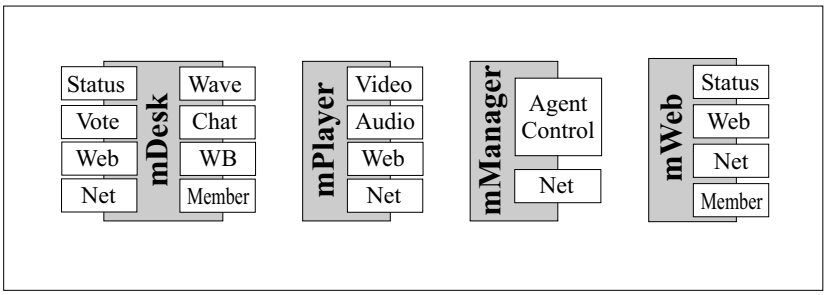

Fig. 1. Some of the mStar applications and their agents. Note, that the same agents are reused in different applications.

The address field consists of four parts: `Host / CB-id / Agent-type / Instance`.

2. The *to section* which identifies the destination of the message. The address format is the same as for the from section with the exception that any part can be replaced with a wildcard. This allows for simple and dynamic addressing of messages where a message can be directed to a single agent, all applications within a session or a group of agents independently of which application they reside in.
3. The *message id section* which together with the from section forms a unique identifier within the session. This section consist of a sequence number that is increased for each transmitted message.
4. The *message section* which contains the message itself.

The traffic on the CB can be both one-way and two-ways, i.e., messages can be of the information-type where an agent is announcing something and is not expecting any response or it can be a request for more information where the requesting agent is expecting a response.

To simplify the protocol only text based arguments and values are allowed. If a specific application needs another type, then that application has to take care of conversion to and from a text representation. The use of a simple and pure textual message format makes it easier to create a platform independent messaging system. It also makes it easier to integrate the control bus into existing applications. As the target of this work is to create a framework that should support developers in the process of creating new distributed applications, the choice of using a text based protocol also makes it easier for developers to debug their applications and make them more stable.

Of course, there is a cost of using text based messages, both in processing and in the amount of bandwidth used but remember that the focus here is to support the transportation of small information reports and occasional control messages. Therefore, it is argued that the overhead in converting the messages between application specific formats and their corresponding textual CB representation can be neglected. For an evaluation of the amount of bandwidth needed for control and management, see section II-D below.

B.1 Resource Discovery and Messaging

Until now, it have been discussed how to get information from applications/agents and how to address messages. We now continue with discussing the resource discovery and actual basic messages needed in the framework. There are two aspects of resource discovery, finding the actual agents to control and finding out what control messages they support.

An important aspect is to be able to find which agents are available within a certain group. Administrators should not have to keep track of which users use which applications and when. This process is supported by the `alive` message which requests all agents to return a short answer if they exist and the `info-request describe` message for retrieving a textual description of the agent. This allows the administrator to dynamically build a list over currently active agents.

In difference from SNMP we propose that every controllable agent should be able to supply information about which variables are controllable in that particular agent. This leads to a dynamic environment where every agent can evolve and be further supported with new control functionality as needed. The alternative would be to define all controllable variables and accessible methods in definition documents and very rarely change these. We argue that this creates a stale environment that puts to much demand on the design phase instead of allowing developers to extend the interface as needed and promoting an evolving environment. This discovery process is supported by the `info-request describe-all` message which requests a summary of all supported commands. Each command can be either write-enabled, read-enabled or both. The framework does not separate commands from get/set of variables in an agent. A command invocation is modeled as a set (with or without arguments).

Finally, the control and management framework must contain messages for actually invoking the get and set methods and these are called `get-request` and `set-request`.

Fig. 2 show a number of example messages with corresponding control bus headers. Using these basic messages more information about the agent of interest and information about which methods it supports can easily be found.

C. Automatic Response Control Filters

It is not always a human person can be available to monitor the activity within an organization. To target this, the framework contains the functionality to set up so called *automatic control response filters*. These filters can be set to respond to special conditions within an installation with predefined actions. For instance, if a user suddenly starts transmitting video with a bandwidth above a specific level a control message is sent to that application to lower the allowed bandwidth. These filters can also be set to trigger external programs which allows for easy

```
130.240.194.245/2345/mManager/45663 */2345/net/* 7 Alive
130.240.64.42/2345/net/33664 130.240.194.245/2345/mManager/45663 73 OK 7 Alive
130.240.64.54/2345/net/883243 130.240.194.245/2345/mManager/45663 22 OK 7 Alive
130.240.64.97/2345/net/43526 130.240.194.245/2345/mManager/45663 983 OK 7 Alive

130.240.194.245/2345/mManager/45663 130.240.64.42/2345/net/33664 8
                                                    info-request describe short
130.240.64.42/2345/net/33664 130.240.194.245/2345/mManager/45663 74
                                                 info-reply 8 mStar network agent

130.240.194.245/2345/mManager/45663 */2345/audio/* 9 set-request size-ms 40
130.240.194.245/2345/mManager/45663 */2345/video/* 10 set-request fps 15
```

Fig. 2. Example CB messages. **top**: resource discovery. **middle:** information request. **bottom:** invoking methods in several agents (note, that no answer or acknowledgment is sent as the underlying mechanism is expected to be reliable).

extension of the control framework with installation specific applications.

D. Bandwidth Usage

Table I presents the number of messages and bandwidth consumed for doing a number of control tasks. Column 1: The task being performed (see below); 2: Total number of packets transmitted; 3: Number of bytes consumed without counting the underlying transport protocol; 4: Number of bytes consumed including the transport protocol using the reference implementation (see Sect. IV). Additionally, numbers for the effect of using unicast (top data in each row) versus multicast for transporting the control messages is presented. Packet loss and retransmissions are not considered as that is very network, situation and implementation specific.

Task 1 - Resource discovery: find all video agents in the session. For the unicast case, it is assumed that the resource discovery is done manually and therefore not included here. Packets: n+1 Bytes excl.: 63*1 + n*92. The constants are based on the mean size of normal messages and n is the number of agents in the session.
Task 2 - Information retrieval: get the current bandwidth setting for video transmission from all agents. Packets: (n + n) vs. (n + 1) Bytes excl.: (109*n vs. 80) + n*103
Task 3 - Variable modification: set the bandwidth to be used by the video transmitter in each agent. Packets: n vs. 1 Bytes excl.: 113*n vs. 84

TABLE I
NUMBER OF PACKETS AND BYTES NEEDED FOR A NUMBER OF DIFFERENT CONTROL TASKS USING UNICAST VERSUS MULTICAST IN A SESSION WITH 100 AGENTS.

Task	Number of Packets	Bytes excl. transport	Bytes incl. transport
1	-	-	
	101	9263	12899
2	200	21200	28400
	101	10300	13936
3	100	11300	14900
	1	84	120

It is hard to compare the amount of actually used bandwidth to other systems as it depends very much on the underlying transport mechanism used and therefore we limit ourselves just to compare the usage of unicast vs. multicast which is especially noteworthy. The bandwidth saving by using multicast instead of unicast for controlling agents is in task 2 of the ratio 1:2 and in task 3 as high as 1:n.

E. Delay

Due to the bandwidth control in the framework where agents calculate a delay before sending messages, there is an total delay before a task can be completed. This delay is dependent on several different parameters such as the number of agents in a session (collected from the underlying transport protocol if available otherwise based on earlier 'alive'-tasks), the available bandwidth (usually depends on the IP-TTL/scope of the session) and the average size of the messages being sent including packet headers. The delay is calculated as shown in (1).

$$\text{delay} = \frac{\text{Rand(0,1)} * 8 * \text{size} * \text{n}}{0.5 * \text{BW(TTL)} * 1024} \quad (1)$$

Table II presents the approximate delay for task 2 and 3 presented above. The network transport delay is not included here as that depends on the network setup and geographical conditions. The table shows that the delay gets long for large sessions with a large scope (e.g., about 8 minutes for 10000 agents in an global session). The reason for this is the very restrictive bandwidth allocation for global sessions in the current implementation. Note, that the exact amount of bandwidth to be used is implementation and session specific and depends on the underlying transport mechanism. For sessions with smaller scopes the delay is acceptable and usable even with up to 10000 agents (5-15 seconds to complete a message transfer).

F. Security and Privacy Issues

It is important that not just *any* user that have access to the control and management framework can control any other application. This is a privilege that should be reserved to authorized users.

Within the control and management framework this is currently solved using public/secret key based digital signatures [18]. Public/secret key technology means that each key is divided into a public and a secret key-part with the convenient

TABLE II

THE DELAY FOR n AGENTS TO SEND A SIMULTANEOUS MESSAGE.

Number of agents (n)	TTL 1-16 (ms)	TTL 17-64 (ms)	TTL 65-128 (ms)	TTL 128-255 (ms)
10	5	15	120	483
50	25	75	604	2416
100	50	151	1208	4833
1000	503	1510	12084	48339
10000	5035	15106	120849	483398

property that the respective part cannot be calculated from the other. If a user encrypts something with the public part it can only be decrypted using the corresponding secret part. This can be used for digital signatures by calculating a digest (a unique digital summary of the data) and then encrypting it using the secret key. Anyone that have access to the public key can then decrypt the signature and compare the resulting digest to a digest that is calculated locally on the data in question. Note, that a correct signature can only be generated by using the correct secret key.

A predefined public key is stored in each client application and when a control message is received the optional accompanying digital signature is verified against the predefined public key. If the signature matches, the command is carried through and if it does not, a warning message is sent to the group warning about that there might be a vicious user trying to break the application control scheme. The use of digital signatures allows the messages to be sent in plain text and even if a message and its signature is snooped on the network, the snooper cannot do any harm within the session as s/he does not have the corresponding secret key needed to generate new signatures.

The predefined public key can be specified in the installation program for automatic handling within an organization. An optional override public key can also be specified and can be used to change the normal operational key if its corresponding secret has been compromised. This override key should of course be stored in a secure way and is only to be used in very rare cases.

The same digital signatures can be used for enabling privacy of reports if needed by encrypting the reported data using the public key. This means that only the users that have access to the corresponding secret key can decrypt the report and view its contents.

III. USAGE SCENARIOS

This section presents a number of usage scenarios of how the management and control environment can be and is being used.

A. Session Membership Management

The mStar environment can be used for light-weight desktop conferencing. It allows any user that have access to the application to transmit audio and video within a session.

Using the control and management framework, administrators can get an overview of the members of a session and they can control who within the session should be able to transmit. This can be controlled in real-time on a user by user basis.

B. mWeb Configuration Control

When the mStar Web presentation system[1] was deployed within a large organization (several hundred installations) and during a critical session broadcast a problem with the existing environment was found (due to the organizations special set-up of its computers) where Web slides could not be displayed on users local computers. The software solution was simple and found quickly but the problem was then how to spread the modified software and the new settings to all listeners. If the control and management framework had been in place at that time it would only had been a question of sending out the correct message to all active instances of the application.

C. Better Perceived Quality of Service using Adaptive Applications

The framework can be used to control the maximum allowed bandwidth within a session when it is noticed that the total utilization of the involved networks gets above a specific level. Together with automatic filters the framework can be used for creating a *better perceived Quality of Service environment* where a control process allocates bandwidth to session as needed and control applications so they cannot utilize more network bandwidth than allowed. Earlier this have only been possible using static configurations and control by an operator. Now it can be controlled dynamically in real-time depending on other concurrent sessions. This creates a set of adaptive applications that better utilizes the available bandwidth and scales to a larger number of simultaneous users *and* sessions as over-utilization by a greedy user/application can be prevented.

Imagine a network setup with four hosts where host A, B and C are running multimedia applications and X is the controller. The total bandwidth is set to 400 Kbps for all sessions on the network. Host A belong to session 1 and hosts B and C belong to session 2. Session 1 has 25% (max 100 Kbps) of the total bandwidth allocated to it. Initially only host A is active and it is transmitting with a bandwidth of 600 Kbps which is above the allowed maximum. The controller gets information about this via the regular reports and changes the transmit bandwidth of A to 400 (as nobody else is currently transmitting, A gets all available bandwidth allocated to it) Kbps via a control bus message. Next B and C joins and both start transmitting at the same time. The controller now lowers the bandwidth of A to 100 and sets the maximum allowed bandwidth for B and C to 150 Kbps each. After a short while the controller notices that host B is only transmitting with a bandwidth of 50 Kbps (the user selected to only transmit with this bandwidth) and therefore increases the available bandwidth for host C to 250 Kbps. When host A later leaves the session the controller reallocates the bandwidth from session 1 to session 2 and sets the maximum allowed bandwidth of C to 350 Kbps (note, that host B is still only transmitting with a bandwidth of 50 Kbps).

The framework can easily be used by applications to retrieve information about the currently allowed maximum bandwidth when the application starts. Instead of building this logic for adaptive applications into the applications themselves it can eas-

[1] A system called mWeb for distributing Web based presentations using IP-multicast [19].

ier be controlled in real-time what policy should be used at a specific moment and installation.

D. Better Group Awareness

The mStar environment can be used as an *electronic corridor* where every user that wants[2] transmits a low bit-rate video stream from their office. This allows for their colleagues to peak into their offices and get a view of what they are currently doing. This can be used for seeing if the other party is available or not before trying to contact him/her. It can also be used to create better awareness in a distributed group environment where users geographically separated (e.g., working in different offices or from their homes) can see if their colleagues are working as well. In the existing mStar system every user sends the amount of video they decide themselves, independently if anyone is actually viewing the video-stream or not.

Using the control framework the receiving mStar instance should instead notify the other party's application when the local user is looking on a particular stream. When the other, remote application gets notified it should increase the amount of bandwidth used for that particular video stream. Further, the more users viewing the same video stream the higher bandwidth should be allocated to that stream.

The same concept can of course be used for real-time electronic meetings and lectures where the current number of viewers controls the relative bandwidth of a specific video stream. Note, that this example differs in its architecture from the earlier one in that it does not have any manager process controlling the session but instead all control is handled within the session itself.

IV. Implementation and Future Work

A reference implementation of the control and management framework have been implemented using the Java programming language and is currently deployed in several different products and applications that are available on the market.

A graphical front end and example control application called the *multicast Manager - mManager* have been designed and constructed. It allows a user to retrieve information about currently running applications and control these applications. For an application to be controllable by the mManager it has to contain the control bus software.

As the whole framework is IP-multicast based an mManager can be started on any host in the network and any number of simultaneous instances can be running at the same time. This allows for flexibility for the administrators as they will not be bound to any specific control station which is normally the case in large systems.

The underlying transport protocol for reliable multicast currently being used is SRRTP [4] which is an extension of the Real-time Transfer Protocol [16] for reliable transfer of data. It is based on the ideas from the SRM framework [20] with the key features of all members in a session helping out with repairs and heartbeats from senders together with negative acknowledgments are used to detect and signal packets loss.

The programmatic interface to the control bus is a simple interface where each agent registers which methods it wants to export. The registration includes a textual description of what the access method does, how many arguments it takes and which local methods should be called on get/set requests.

The security parts discussed earlier are also implemented using the Java standard libraries for key management and cryptography.

A. Future Work

An open user interface issue is how changes to an active instance of a program should be presented to the user and more precisely which changes are significant to notify the user about. For instance, if a controller requests an application to lower the bandwidth for its outgoing video stream by 1 Kbps. Should this cause the user to be notified or not? In this case it depends on the current bandwidth being used. In the current implementation of the framework it is left to the application to decide which changes should result in the user being notified.

The security system is still in its infancy and have to be further investigated. Specially the interface to the agent of how to specify who has access to which methods and information have to be further investigated and defined. This is a delicate problem as programmers want a simple but at the same time powerful interface.

The current automatic response filters are very simple and limited due to the lack of a good and simple action specification method. Initial interfaces of interest are SMAP [7] and the autonomous agent services of the Java Dynamic Management Kit [21]. Note, that the target is still to create an Java independent framework, i.e. one that is not dependent of the Java runtime environment.

With the framework in place it also would be interesting to pursue the issue of software updates by distributing software components using IP-multicast for dynamic on-demand update of software and permanent changes to software installations.

V. Conclusions and Summary

In the introduction we set out a number of requirements that the framework have to support. This section evaluates how these requirements have been addressed in the framework design.

- *The framework have to support large groups of applications, both for reporting and control.* This have been addressed by using IP-multicast for both control and reporting. Also, the delay between messages is dynamically calculated based on the current number of members in a session and the currently available bandwidth.
- *The framework have to support efficient control of groups of applications without the need to send control messages to each application explicitly.* The control bus allows single messages to be addressed to more than one application/agent.
- *The framework have to protect users from unauthorized control of their applications.* This have been addressed by including support for asymmetric cryptography and digital signatures.
- *The framework have to allow developers to easily insert control access points in their software.* This have only been ad-

[2] It is outside the scope of this paper to discuss why users would want to send video from their offices but experience have shown that camera shyness can be overcome after the added value to a distributed working environment is understood and that the purpose is *not* for the boss to monitor the employees but to support the daily teamwork.

dressed from a Java perspective where a simple but powerful API have been defined.

- *The framework should be as independent of the underlying software and transport technology as possible.* The whole framework is independent of the underlying transport mechanism but requires IP-multicast for scalability.
- *The framework should allow for scalable resource discovery of both available control objects and controllable variables and methods in these objects.* This have been addressed by allowing controllable agents to announce themselves and provide information about available control points.

This paper presents a framework for control and management of software applications. It allows applications to distribute messages in a scalable way, with regard to both the number of applications currently running and the available control bandwidth. This is done using IP-multicast, a messaging platform called the Control Bus (CB) and a reliable multicast protocol (SRRTP). Note, that the whole framework is designed without any special requirements from the underlying transport mechanism as long as it is transport reliable and uses IP-multicast (unicast can be used but the framework becomes much less scalable then).

The novel usage of IP-multicast for management and control creates a mobile and scalable framework that can be used for a number of different applications. We presented an evaluation of the bandwidth usage and the total delay for doing control tasks.

To utilize the control and management framework and administrate running applications, a program called the multicast Manager - mManager have been developed. The mManager gives the administrator an overview over currently running applications and their agents. It also allows administrators to control these applications.

We presented how the framework can be used for creating better group awareness in a distributed real-time environment. Further, it was presented how the environment can be used to create a better perceived Quality-of-Service environment for distributed media applications where bandwidth is dynamically allocated between active sessions and the total amount of used bandwidth is controlled automatically by a control process. These different usage scenarios show that the framework is flexible and can be used for a number of different applications.

The framework is currently being evaluated in the industry so it is too early to draw any final conclusions about its practical usability but initial tests show that it works and is powerful enough to be used in real deployed application in the industry (note, that the CB and the SRRTP have both been used in large scale earlier in other contexts and applications for several years and should therefore be considered as stable).

The work presented in this paper is based on requests from both academia and industry. We believe that this framework simplifies the administration and control of large groups of distributed applications and real-time multimedia sessions.

Acknowledgments

Thanks to Claes Ågren and Serge Lachapelle (Marratech AB), the people at Ericsson Data and the anonymous reviewers for comments, suggestions, and feedback.

This work is supported by the Centre for Distance-spanning Technology (CDT), Marratech AB and the Swedish National Board for Industrial and Technical Development, NUTEK.

REFERENCES

[1] S. E. Deering, *Multicast Routing in a Datagram Internetwork*, Ph.D. thesis, Stanford University, 1991.
[2] "IPMI - the IP-Multicast Initiative web site," [3].
[3] P. Parnes, *The mStar Environment - Scalable Distributed Teamwork using IP Multicast*, Luleå University of Technology, September 1997, Licentiate of Engineering Thesis.
[4] P. Parnes, "Scalable Reliable Real-time Transport Protocol - SRRTP," Work in progress[4], 1996.
[5] "Protocol operations for version 2 of the Simple Network Management Protocol (SNMPv2)," IETF RFC1905.
[6] W. Richard Stevens, *TCP/IP Illustrated*, vol. 1, Addison-Wesley, 1996, chapter 25.
[7] Y. Izumi, T. Nakai, S. Yamaguchi, and Y. Oie, "Design and implementation of an agent system for application service management," in *Proceedings of ISOC INET'98*, 1998.
[8] M. Handley, I. Wakeman, and J. Crowcroft, "The Conference Control Channel Protocol - CCCP: A scalabe base for building control applications," in *ACM SIGCOMM*, 1995.
[9] S. McCanne and V. Jacobson, "vic: A flexible framework for packet video," in *Proceedings of ACM Multimedia*, 1995.
[10] Mark Colan, "Infobus 1.1 specification," Tech. Rep., Sun Microsystems Inc. and Lotus Development Corp., March 1998, [5].
[11] "JavaSpace specification," March 1998, [6].
[12] Silvano Maffeis, "iBus - the Java Intranet Software Bus," Tech. Rep., SoftWired AG, February 1997, [7].
[13] "The OMG - Object Management Group web site," [8].
[14] ISO/IEC JTC 1/SC 29 N 2206, "Information technology - Generic coding of moving pictures and associated audio information - Part 6: Delivery Multimedia Integration Framework," 1998.
[15] "Common Channel Signaling System No. 7 - SS7," ITU Standard.
[16] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," 1996, IETF RFC1889.
[17] P. Parnes, "Control bus specification," Work in progress[9], 1996.
[18] W. Stallings, *Network and Internetwork Security - Principles and Practice*, Prentice Hall, 1995.
[19] P. Parnes, M. Mattsson, K. Synnes, and D. Schefström, "The mWeb presentation framework.," *Computer Networks and ISDN Systems*, September 1997.
[20] S. Floyd, V. Jacobson, S. McCanne, C. Liu, and L. Zhang, "A reliable multicast framework for light-weight sessions and application framing," in *ACM SIGCOMM*, 1995.
[21] "Java Distributed Management Kit 2.0," 1998, [10].

[3] <URL:http://www.ipmulticast.com/>
[4] <URL:http://www.cdt.luth.se/~peppar/docs/rtp_srm/>
[5] <URL:http://java.sun.com/beans/infobus/infobus-1.1.pdf>
[6] <URL:http://java.sun.com/products/javaspaces/specs/js.pdf>
[7] <URL:http://www.softwired.ch/ibus/ibus_overview.pdf>
[8] <URL:http://www.omg.org/>
[9] <URL:http://www.cdt.luth.se/~peppar/docs/>
[10] <URL:http://wwwwswest2.sun.com/software/java-dynamic/tech-overview.html>