The ITB Journal

Volume 4 | Issue 2 Article 9

2003

Towards a Framework for Modelling Multimedia Conferencing Calls in the Next Generation Network

Gavin Byrne

Declan Barber

Towards a Framework for Modelling Multimedia Conferencing Calls in the Next Generation Network

Gavin Byrne and Declan Barber

Institute of Technology Blanchardstown

gavin.byrne@itb.ie declan.barber@itb.ie

Abstract

This paper is concerned with the creation of a multiparty multimedia conferencing application which can be used in Next Generation Networks. It begins by suggesting ways in which conferencing can be modeled with a focus on separating signaling and media transfer functionality. Enabling technologies which could support the modeling framework derived and which are compatible with Next Generation Network (NGN) principles are reviewed. Finally, a design and implementation for a simple multimedia conferencing application are described.

Keywords

Multiparty conferencing, Multicast, SIP, RTP, Java, JAIN, collaboration

Introduction

Multiparty conferences over the Internet are increasingly capable of providing real-time media distribution (voice or video) rather than the non or near real-time functionality (message board, chat room) of the past One reason for is the increasing availability of broadband access and the introduction of Universal Mobile Telecommunications Systems (UMTS), which offers considerably more bandwidth than Global Systems for Mobile Communications (GSM). This increase in bandwidth will be a key factor in the increased use of media rich real-time conferencing. Another reason is the increasing support for real-time communication provided by Internet Protocols such as RTP/RTCP and SIP. This article represents an approach taken in the early stages of an applied project in ITB that is focused on researching the potential for creating new real-time multimedia conferencing services in the NGN. Potential applications areas include education, emergency response services, gaming and any general collaborative application.

This paper makes certain assumption based on earlier work [1,2,3] that can be summarised as follows:

- Convergence in the NGN will be based firmly on open standards and the TCP/IP protocol stack in particular
- Applications leveraging existing and emerging Internet Protocols will dominate

- Bandwidth availability will steadily increase from end-to-end and decreasingly represent a technical constraint

Part I: Multiparty Conferencing Models

It is possible to identify differing models for multiparty conferencing based on the physical topology, media (type, bandwidth usage and heterogeneity), logical connectivity, network-layer delivery, the distribution of intelligence within the network, signaling and application. There are inter-relationships between these issues. Assuming that IP will provide best effort delivery in the NGN over an essentially transparent data-link, we will focus our discussion on conferencing models to the issues of topologies (and the distribution of intelligence within the topology), signaling, media transfer and higher layer protocols. Aspects of lower layer functionality such as bandwidth usage and differing codecs for speech/video will be assumed to have been resolved by the trend towards convergence.

Topology and the Partitioning of Intelligence

Multiparty Conferencing can use a centralized (Star/Hub & Spoke), distributed (Full Mesh/Partial Mesh) or hybrid (Partial Mesh/Tree/Extended Star) topology.

Centralised: In the centralized model, a central server receives the media streams from different participants, combines them and redistributes them as needed. This model emphasizes the placement of intelligence (for combining, codec translation and redistribution) at the central processing node and means the end-user nodes can be relatively simple. Other important advantages are the relative ease with which conference participants with different media codecs can be supported and the ability to track participants and manage the conference. The obvious disadvantage is the reliance on the central node for conferencing.

Fully Distributed: In a distributed topology, each end node sends a copy of its media stream to all other participating end nodes. This requires each end node to have sufficient intelligence and processing power to translate codings, sum incoming media streams and negotiate and manage participation. It also adds complexity in the event of people joining or leaving the conference on an ad hoc basis.

Hybrid: The hybrid model combines some of the benefits of both the centralized and distributed models, requiring intelligence in both the central and end-user nodes. It behaves like the central model insofar as some media or signaling streams are sent to the central node but the central node is only required to re-distribute the incoming streams. There is no need to

centrally mix or filter the streams before redistribution. Interim nodes such as gateways can provide codec translation between SIP and H.323, GSM, ISDN or other codings.

Network Delivery

It is possible to deliver media for conferencing using IP Unicast, Broadcast, Multicast or any combination of these. We summarily dismiss broadcast for media transfer because of its impact on connected host networks, bandwidth and the lack of support for broadcast in IPv6. Multicast trees offer clear advantages in terms of bandwidth usage and scalability for conferencing, as the media stream is only replicated once for each subscribed branch of the multicast routed tree. This use of bandwidth can be further optimized with multicast-aware OSI model layer 2 switching. Multicasting is therefore ideal for conferencing applications both on the LAN and the wider internetwork. While easily achieve on the enterprise LAN, however, native multicast capabilities are still not widespread on the Internet and this is a limiting constraint on existing approaches to multiparty conferencing. Unicast works well for a small number of conference participants but does not scale well for a large number of users. It is possible, however, to use combine unicast and multicast in an effective manner where the number of speakers is low e.g unicast could be used for sending media streams to a central node while the conference output stream can be redistributed using multicast to the participating end-nodes. In this way, any participant node in a given conference merely subscribes to the corresponding multicast address. Speaking generally, this hybrid will scale best (i.e. the disadvantages of unicast for transmission to the central node will be mitigated) if the number of conference speakers is small while the number of listeners is large.

Signaling for Multiparty Conferencing

Signaling refers to the exchange of information between call components required to provide and maintain service between two or more endpoints. This is achieved by exchanging protocol specific messages. Signaling messages carry information related to the following:

- capabilities exchange
- opening and closing of logical channels used to carry media streams
- flow-control messages
- general commands and indications

Signaling is a critical mechanism for call setup and service delivery in conferencing. Signaling protocols make it possible to establish point-to-point and point-to-multipoint links over a converged network architecture that can span TCP/IP LANS, the Internet, traditional WANS (PSTN, ISDN, FR), etc. With this link established it will not only be possible to send voice and video, but any IP based packet data like multimedia presentations, still images, text, etc. The

differences in the signaling protocols that have emerged arise largely from the different origins and philisophical approaches that spawned them and can be summarised as:

Intelligent Network Approach (Centralised): is the traditional approach of the telecommunications industry and assumes that the network is intelligent and the end nodes are dumb e.g. MGCP (Media gateway Control Protocol), H.248/Megaco, etc. These protocols are highly complex, they don't fit the Internet model and are not directly compatible with existing LAN infrastructures.

Intelligent Node Approach (Distributed): The end nodes are intelligent but the network is dumb e.g. the Session Initiation Protocol (SIP) which is designed for use over the Internet. SIP has low complexity, is designed for the Internet and provides a simple clean end-to-end architecture.

Intelligent Nodes & Network Approach (Hybrid): Both the network and the end nodes are intelligent e.g. H.323 which was designed for use over the enterprise LANs. Although the best established protocol for multimedia conferencing on the enterprise LAN, H.323 is highly complex, not very scalable, doesn't fit the internet model and is expensive to deploy.

The figure below summarises the protocol stacks used with each of the signalling protocols.

Media Transfer

The Real-Time Protocol (RTP) and the Real-Time Control Protocol (RTCP) are the main IETF protocols for transferring media in real-time over the Internet. The User datagram Protocol (UDP) is used at the transport layer because of the reliability provided by RTCP. RTP/RTCP packets can in turn be delivered using unicast, multicast or broadcast addressing. The fact that SIP separates signaling from media transfer is an illustration of an important modeling concept in conferencing, namely that it is possible to separate the design of the signaling from the design of the media transfer. Indeed, different aspects of signaling and different aspects of media transfer could be handled separately, allowing a more granular approach to the design of these aspects of conferencing.

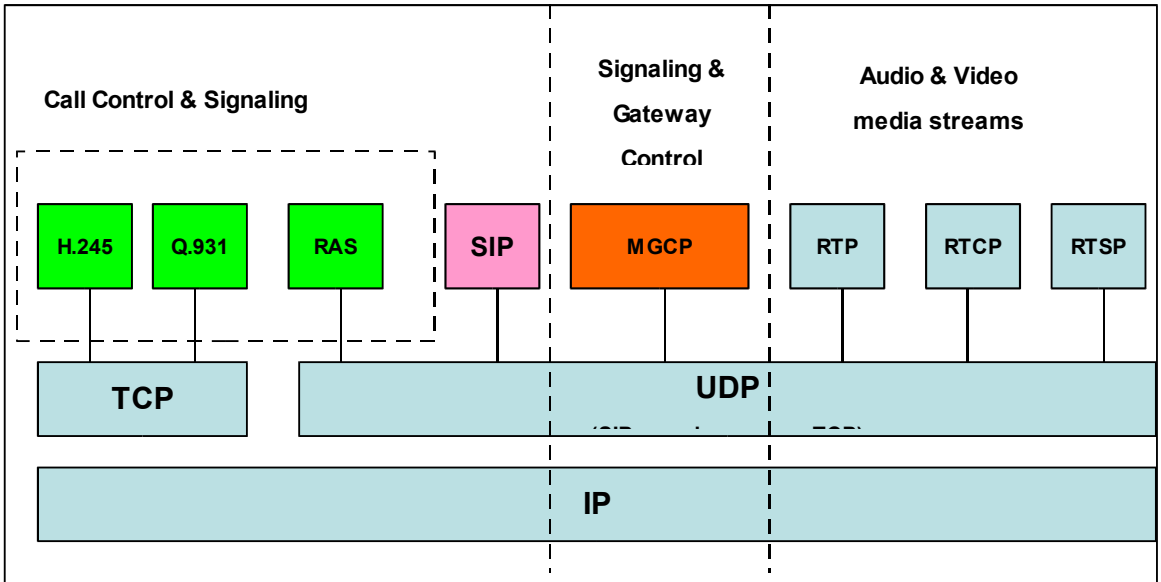


Figure 1: Comparison of Signaling Protocol Stacks

Application

Applications may vary according to the conference size (number of participants), profile (ratio of speakers to listeners, open or closed), media type (text, audio, video or combined), environment (LAN, WAN, Internet or combination) and the end-node profile (access bandwidths, codecs, user interfaces and available protocols). The ratio of speakers to listeners is an important aspect of the conference system design. Multicast will scale well for listeners but not necessarily as well for speakers, especially in a shared or non-intelligent layer-2 switched environment. So the number of speakers impacts more directly on scalability. The conference media type will directly impact on required bandwidths while its open or closed requirements will add management overhead. The capabilities of the end nodes in terms of access bandwidth, employed codecs, interfaces and available protocols are also an influencing factor in system design. The Application goals and constraints will represent the main drivers in design of a conferencing system solution and it is difficult to see how any approach other than a structured top-down approach could be adopted.

Towards a Framework for Conference Modeling

One framework for modeling conferencing systems that may be useful is to consider the following design issues separately:

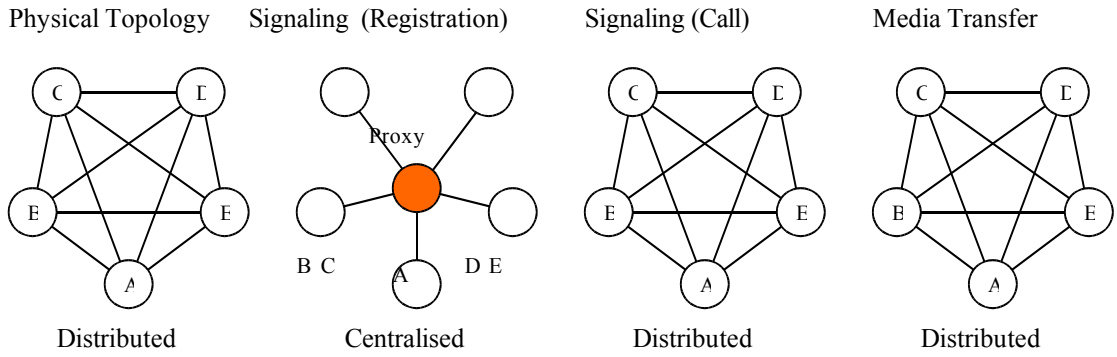
- Physical topology/Logical Connectivty
- Packet Delivery
- Signaling (Registration, Naming and Location)
- Signaling (Call establishment, maintenance/manipulation and termination)

- Media Transfer
- Application

This framework model is clearly consistent with the existing OSI and TCP/IP models. Not only could it be applied separately to the signaling and media transfer function, to could also be applied to different aspects of the signaling function (e.g. registration and calling) or of the media transfer function (e.g. for incoming and outgoing media streams). Centralised structures imply that unicast or broadcast (from central node) are possible for delivery. Distributed structures imply multicast is also an option. The following Figures illustrate the use of such a modeling framework to develop some basic conferencing models.

Example 1: In this fully meshed example, each participant in the conference has a direct connection with each other participant. Signaling could be provided separately using a centralized model for registration and address resolution functions while calls could be established directly between peers. In the full mesh, any node could function as the registration proxy or it could be a separate node. User A would have 4 incoming streams, and 4 outgoing streams. That is 8 media streams to send over one link and to process with one device. Straight away it is clear to see that this architecture is not very scalable all and would be appropriate only for conferences with a small number of participants.

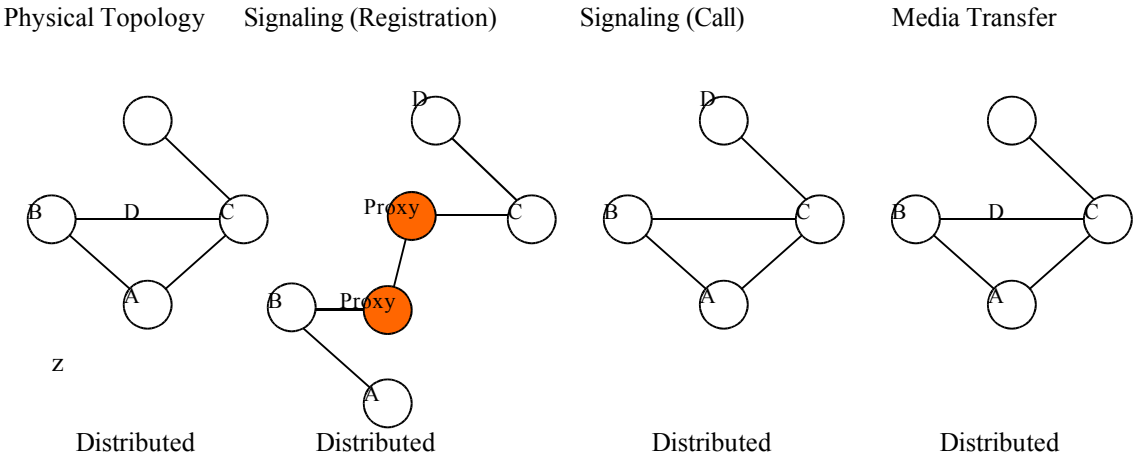
Figure 2: Fully Meshed Conference Architecture



Example 2: In a partially meshed architecture, some users can connect to each other directly but others can only see each other through indirect connections. In signaling terms, each node must be able to directly connect to a proxy however or initial registration/location is not possible. A has a connection to both B and C, but B and C only have a single connection to A. C now calls D and brings it into the conference. D receives its media stream through C regardless of the source, so forwarding is a key function which the partial mesh model needs and which was not needed by the full-meshed model. This architecture scales better than the

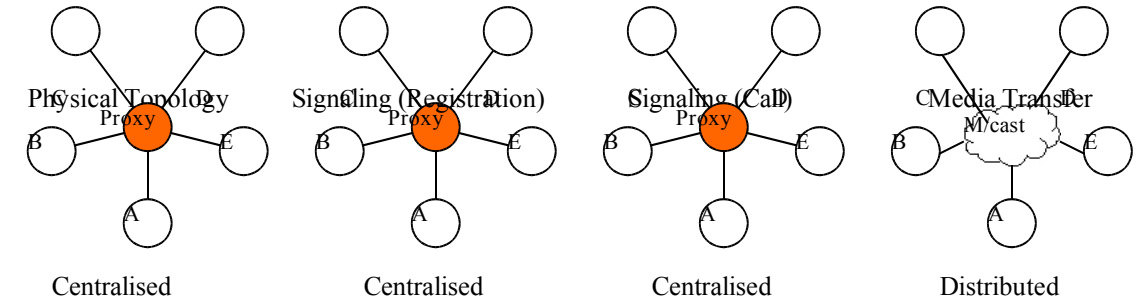
fully meshed and would suit small to medium sized conferences, but again would not scale easily to a large conference. This architecture would most likely arise in the form of an 'ad-hoc' conference where A and B are in a call, and A decides to invite C. In this example, signaling is achieved by one or more centralized proxies. Each node must be in direct connection with a proxy in order to register initially.

Figure 3: Partially Meshed Conference Architecture



Example 3: A more centralized approach is shown in this example. A conference server approach uses an architecture where a central server maintains a signaling dialogue and media transfer dialogue with each participant. For conferencing purposes, the nodes can only send signaling and media to the conference server node (although this does not preclude them from other peer-to-peer sessions with other nodes if the topology allows). The server plays the role of the centralized manager of the conference. Signaling could be unicast and media transfer could be either unicast or multicast, depending on whether it is the incoming or outgoing stream.

Figure 4: Conference Server Architecture



Part II: Enabling Technologies

Signaling

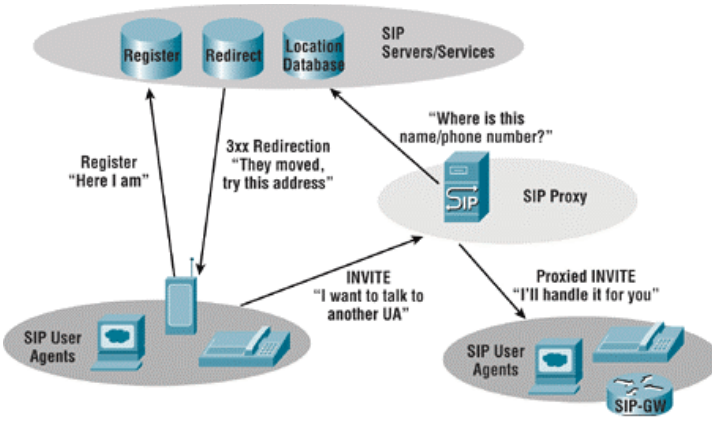
SIP is an application layer signalling protocol which provides call set-up, modification and termination. It can also provide services such as name translation, user location, feature negotiation and call participant management.

We believe that SIP will be the protocol of choice for Next generation Networks and we have chosen SIP to develop our multimedia conferencing application because of its ease integration with existing IETF protocols, simplicity, mobility, scalability, and ease of development, extensibility and deployment in the core and at the edge of the enterprise and support for multicast, unicast or a combination of both. It is designed for integration with existing IETF protocols, using existing protocols and extensions to provide message formatting (HTTP), media (RTP), name resolution and mobility (DNS and DHCP) and multimedia (MIME).

The main SIP protocol components are User Agents (end system) and Network Servers. Each Agent has a Client (caller) and Server (receiver) component. Examples of SIP Network Servers are Registration Server, Redirect Server, Voicemail Server, etc. These Servers can be separate devices in logical terms, but be physically implemented on the one network device, or alternatively can be physically distributed over multiple devices to provide greater scalability. These servers can be stateful (knows the state of all current calls), or stateless (doesn't track calls). SIP provides its own reliability mechanism so it runs over UDP. Importantly, participants can communicate using multicast, unicast or a combination of both.

As a protocol used in a distributed architecture, SIP allows you to build large-scale networks that are scalable, resilient, and redundant. It provides mechanisms for interconnecting with other VoIP networks and for adding intelligence and new features on either the endpoints or the SIP proxy/redirect servers. Each signaling protocol follows this general idea, but each protocol's implementation of signaling varies. The diagram below (International Engineering Consortium/Cisco) shows an example of SIP architecture.

Figure 5: SIP Architecture



As an application layer signaling protocol used in a distributed architecture, SIP is best suited to meet our scalability, real-time, simplicity and extensibility design requirements.

The Session Description Protocol is used in conjunction with SIP for exchanging session capabilities (ability to send and/or receive audio or video, supported codecs, IP address to send media, etc.).

Media Transmission

The Real-time Transmission Protocol (RTP) was defined by the IETF and is used for the delivery of time-sensitive data (e.g. voice, video). As retransmission of lost, or out of sequence, packets is in reality pointless for this kind of time-sensitive data, RTP uses the User Datagram Protocol (Postel, 1980) which has a 'best effort' approach. UDP also has a much lower protocol overhead than the connection oriented Transmission Control Protocol (TCP) (Postel, 1981), which is important for efficiency reasons.

The functionality of RTP is simple. The data to send is divided up into smaller parts, to which an RTP header is added. This header includes information such as the sequence number, a timestamp, and a header which identifies the type of payload. RTP is not able to prevent jitter but it provides enough parameters to compensate for its effects. In fact it is the Real-time Transport Control Protocol (RTCP) which enables the senders and receivers to adapt their sending rates and buffer sizes. RTCP has to be supported by RTP devices in any case. It is suggested that the proportional relation of RTCP in RTP traffic should not exceed 5 percent (I. Miladinovic and J. Stadler).

JAVA support for conferencing applications

This next section focuses on the Java programming language and how it is enabling the development of computer telephony applications which leverage the functionality of the

previously described protocols. The Java platform is based on the idea that the same software should run on many different kinds of computers, consumer gadgets, and other devices. Java's main strengths are this platform independence (or portability), adaptability, scalability, multithreaded ability, and Object Oriented Design.

Java has historically been viewed inferior to C for real-time applications (such as computer telephony applications) as C is able to talk directly to the native code where Java talks through an interpreter called the Java Virtual Machine (JVM). This JVM translates Java classes to byte code which the underlying operating system can understand (which gives Java's its portability). In certain circumstances, Java can talk directly to native code written in languages such as C. This weakness of Java is slowly becoming less of an obstacle to real-time programming with the development of JIT (Just In Time) controllers and with the emergence of real-time Java.

Java provides a wide range of programming API's for various application functions: networking (net package), multimedia (JMF package), encryption (crypto package), GUI design (awt/swing packages), etc. This range of API's is ever increasing, as both developers create their own API's, and third party vendors create publicly available API's, thus enabling developers to easily and seamlessly add new functionality to their systems.

Java also provides support for a wide variety of Internet protocols such as HTTP (applet/servlet packages), SIP (JAIN Framework), RTP (JMF package), IP (net package), which allow development of inter-networked applications. The Java API's of most importance to this project are:

The Java Network package (java.net.*): Through the java.net package, Java provides the ability to create both unicast and multicast sockets for the transmission and receipt of data. This ability to create multicast sockets will be an advantage in certain circumstances where we are sending identical data to multiple recipients as multicast is far more bandwidth and processor efficient than having to open up multiple unicast sockets for the same data. Multicast is a far more scalable solution than unicast and will be very useful for sending & receiving data in conference calls.

The Java Media Framework (javax.media.*): The Java Media Framework (JMF) is a package for developing multimedia applications with Java. It enables easy integration of audio and video clips into an application from a local file, URL, or a device such as a microphone or

web-cam. JMF also provides the necessary methods for the transmission and receipt of real-time media streams using the Real Time Protocol (RTP) and the Real Time Control Protocol (RTCP), which will obviously be necessary for transmitting audio and video during calls.

The Java Intelligent Network Framework: (JAIN including javax.sip.*, javax.sdp.*): The Java Intelligent Network Framework (JAIN) is a set of Java technology based APIs which enable the rapid development of Next Generation communications-based products and services on the Java platform. By providing a new level of abstraction and associated Java interfaces for service creation across point-to-point, circuit-switched (PSTN, ISDN), packet/cell-switched (X.25, Frame Relay, ATM) networks. JAIN technology enables the integration of Internet (IP) and Intelligent Network (IN) protocols. This is referred to as Integrated Networks. JAIN provides specifications for signaling and network service creation, some of which are Protocol API specifications; others are Application API specifications as shown in the table below:

Figure 6: The JAIN Framework

Protocol API Specifications	Application API Specifications
JAIN TCAP 1.1 *(Final Draft)*	JAIN Call Control 1.1 *(Final Draft)*
JAIN INAP 1.0 *(Final Draft)*	JAIN Coordinations and Transactions *(Final Draft)*
JAIN MGCP 1.0 *(Final Draft)*	JAIN Service Logic Execution Environment (SLEE)
JAIN OAM 1.0 *(Final Draft)*	JAIN Presence and Availability Management (PAM)
JAIN MAP	Java Payment API (JPay)
JAIN MEGACO	JAIN Presence
JAIN SIP 1.0 *(Final Draft)*	JAIN Instant Messaging
SIP API for J2ME	JAIN SIMPLE Instant Messaging
JAIN ENUM	JAIN SIMPLE Presence
JAIN SDP	SIP Servlets 1.0 *(Final Draft)*
	JAIN SIP Lite
	JAIN Service Creation Environment (SCE) - SCML
	JAIN Service Creation Environment (SCE) - Java
	Server API for Mobile Services (SAMS): Messaging

The two specifications we are currently using to implement our conferencing applications are JAIN SIP 1.0 and JAIN SDP. The SIP and SDP protocols have been outlined in the previous section of this paper.

For service creation, the JAIN connectivity management specification was submitted for review. This is a specification that encompasses different layers of interfaces for controlling connectivity in intelligent IP networks. Connectivity management is a collection of services for dynamically providing connectivity with specified QoS (Quality of Service), security (using IPSec), and routing attributes in IP networks. This specification was later withdrawn and has

yet to be replaced with another specification which provides these connectivity management services.

The JAIN initiative brings service portability, convergence, and secure network access to telephony and Internet networks. This will positively alter the current business structure of these networks as follows:

- **Service Portability:** - Write Once, Run Anywhere. Technology development is currently constrained by proprietary interfaces. This increases development cost, time to market, and maintenance requirements. With the JAIN initiative, proprietary interfaces are reshaped to uniform Java interfaces delivering truly portable applications.
- **Network Convergence:** (Integrated Networks) - Any Network. By delivering the facility to allow applications and services to run on PSTN, packet (e.g. IP or ATM) and wireless networks, JAIN technology speeds network convergence. As demand for services over IP rises, new economies of scale are possible as well as more efficient management and greater integration with IT.
- **Secure Network Access** - By enabling applications residing outside the network to directly access network resources and devices to carry out specific actions or functions, a new environment is created for developers and users. The market opportunity for new services is huge when controlled access is provided to the available functionality and intelligence inside the telecommunications networks.

Part III - Current Work

Our current research is in the area of sensor data retrieval, display, and collaborative analysis. As part of this, we are working on a SIP agent which offers telephony functionality (one-to-one calls, call forward, busy here, forward to voicemail on busy/no answer, etc) as well as the ability for users to participate in conference calls with a view to collaboratively analysing sensor data. The following section givse an overview of our analysis & design, our prototype implementation, and an example usage scenario of this system. Our initial simple application assumes a single source for the distribution of all combined conference media to which other participants contribute or listen.

Analysis & Design

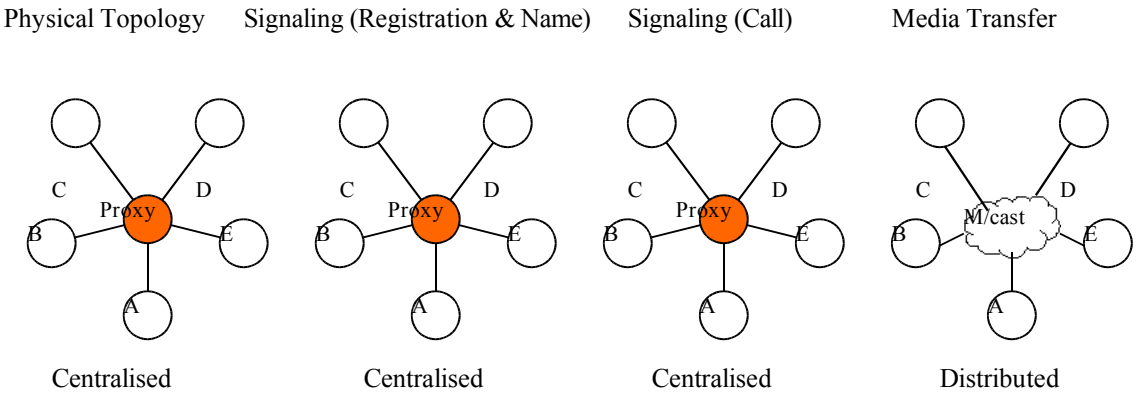
Our conferencing system had the following design requirements:

- Scalability in the number of users (primarily affected by media transfer)

- Efficiency in its use of network and node resources
- Simplicity of implementation
- Extensibility for the rapid deployment of new services
- Real-time Functionality: it must allow effective real-time conferencing

For simplicity and efficiency our approach uses a centralized signaling architecture and hybrid media delivery, where unicast is used for signaling and multicast or a mixture of unicast/multicast is used for media delivery. This is scalable large numbers of participants and bandwidth utilization and processing. Overheads are good. Each participant makes a normal peer-to-peer SIP call to the conference server using unicast signaling. Once the call is established media is sent and received on a multicast connection. The role of the conference server is to act as the centralized manager of the conference, and to maintain a signaling dialog with each participant in the conference.

Figure 7: Conference Server Architecture



Users wishing to join the conference can simply send a standard SIP INVITE message to the conference server which in turn can choose to authenticate the user or simply send an immediate ACK reply to set-up the call. Users currently participating in a conference who would like to invite other users into the conference can send a SIP REFER message with the URI of the conference server, inviting them into the call (this REFER message could alternatively be sent to the conference server with the intended recipient's URI).

Implementation

The SIP agent was implemented by extending the functionality of the basic reference implementation developed by Emil Ivov. These extensions include:

- Converting the application to an applet.

- SIP proxy/registrar registration (the reference implementation was supposed to do this, but it needed some alterations to successfully register to a SIP registration server)
- Call forwarding when busy/unavailable.
- Calls to conference servers using multicast address for audio and video.
- Enabling the applet to identify more SIP messages (e.g. Temporarily Unavailable).
- The ability to view sensor data in graph form which is stored in a web server database, or to collaboratively view these graphs with other called parties (both on-to-one calls and conference calls) and highlight (by drawing on the graph) interesting findings.

The conference server was implemented by altering the SIP web client previously described. The purpose of the conference server is to simply give the conference a focus, or endpoint, to call (and to also store information about collaboration taking place in the conference call allowing users joining the conference to know what everyone is looking at and what has so far been highlighted by others). The conference server does not receive or send any media streams. It simply returns a multicast address to which all users who want to participate can do so to a multicast group to which all other users subscribe.

For collaboration during a conference, the aim was to make the task of collaboration the same for a user irrespective of whether the call was a 2-party call or a conference call. Our implementation involved having a collaboration button which when pressed would send a collaboration request either to the other call party (for a 2-party call) or the conference server (for a conference call). This collaboration processes is shown in the next two diagrams.

Figure 8: 2-party Call Collaboration

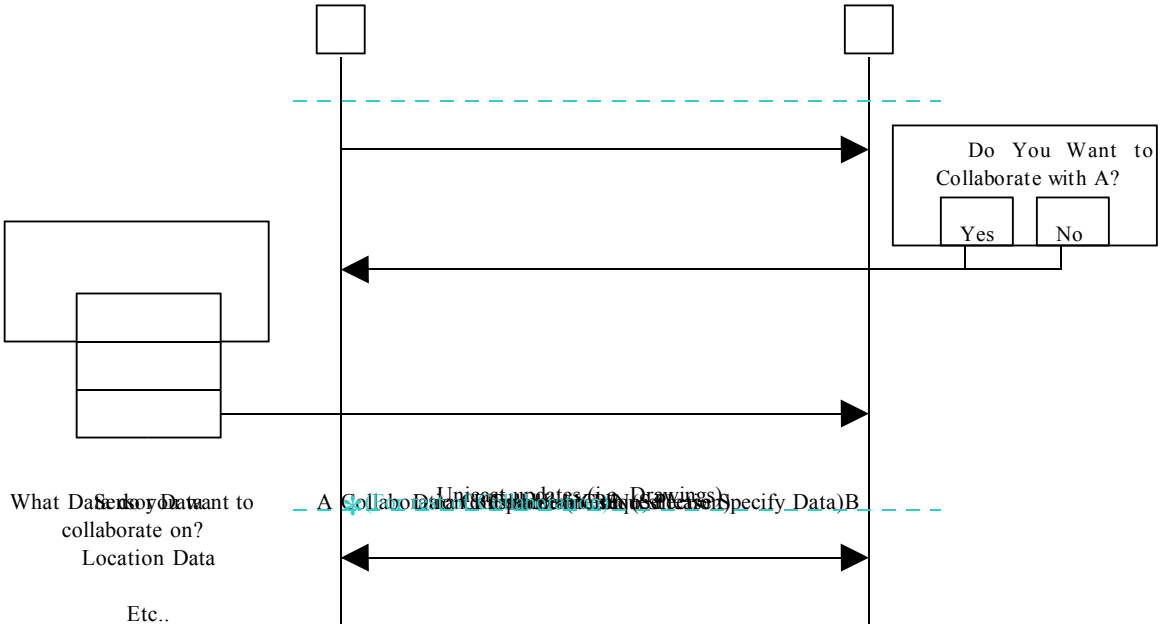
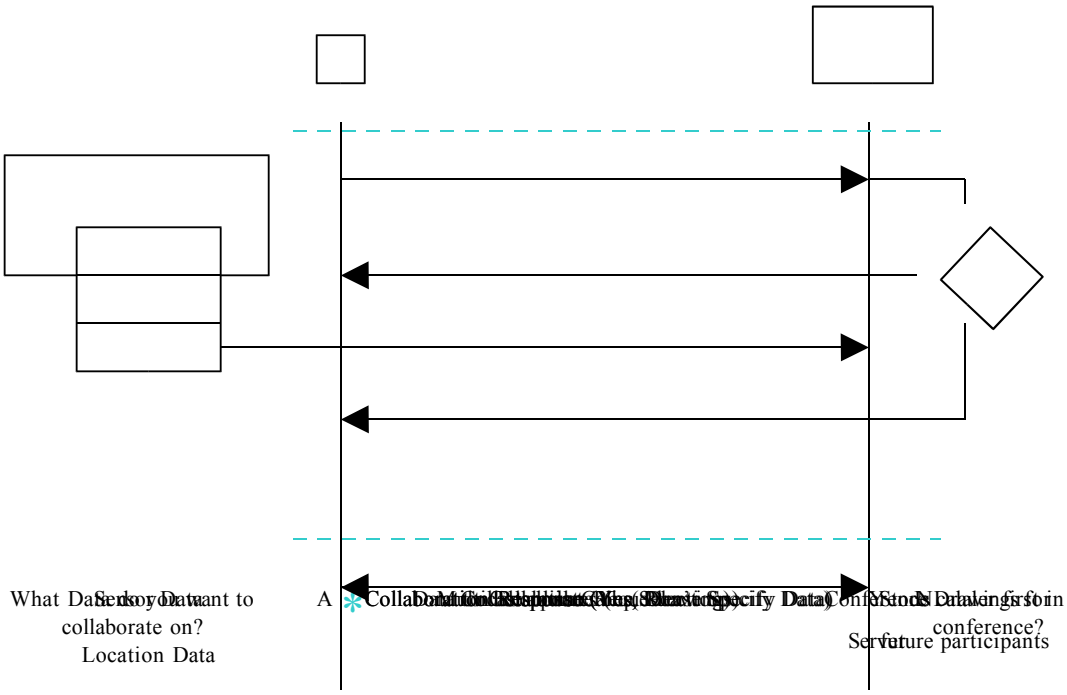


Figure 9: Conference Call Collaboration



Example Scenario

The following example shows a conference call where multiple users want to collaboratively examine sensor data (we will use temperature for this example) from the last 24 hours.

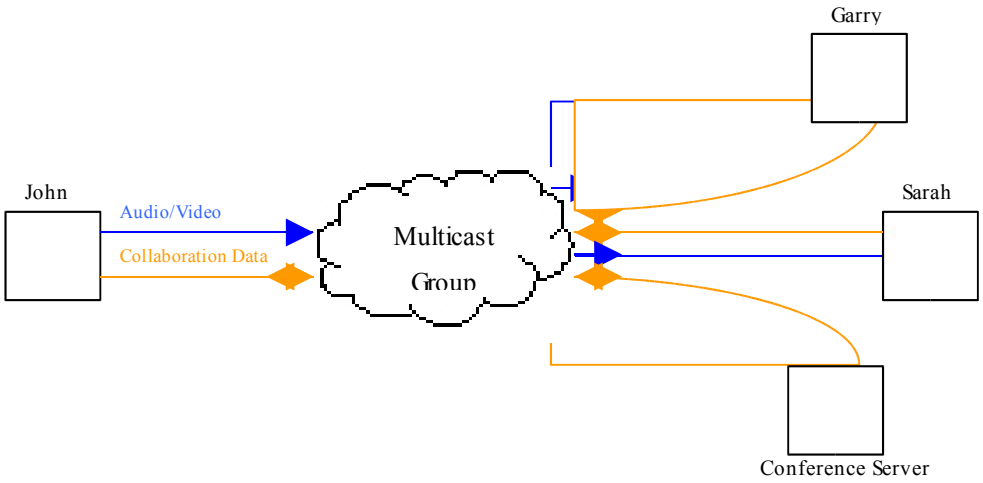
John dials the conference server (e.g. sip:ConferenceServer@domain.com) and is automatically connected. John selects the collaborate option and as he is the first user in the

conference he is asked by the conference server to specify what data to view. John selects the temperature sensor and selects the last 24 hours. John's web client is updated with a graph showing this sensor data. John circles a few parts of the graph which are of interest with his draw tool which are multicast to other participants of the call. Currently there are no other participants in the conference, but the conference server also listens for these multicast updates to store for any future conference participants.

Garry now dials and connects to the same conference server as John. John and Garry can now select if they want to send audio or video (if they have a web-cam), or whether they just want to listen to other users of the conference. Garry now selects the collaborate option and his screen is automatically updated with the same graph John is viewing, as well as the drawings John had made prior to Garry joining the conference. Any drawings Garry makes will be multicast to other participants of the call (John and the conference server).

Sarah now joins the conference and repeats the same process. We now have a three member conference scenario where each user can choose to send or just receive audio and video, as well as collaborate on data stored on a web server. As all this media is sent using multicast packets, this architecture can scale very well, especially if not all users need to be sending audio and video streams.

Figure 8: Media Stream Transmission



The example diagram (figure 8) shows the basic network infrastructure of the conference. In this example only one conference member is talking and the other users (2 in this example) are listening, but all users are collaboratively analyzing the data graph. Any updates being sent to other participants of the conference are also sent to the conference server.

Conclusions

A range of models is possible for designing multiparty conferencing for the NGN. A modeling framework that supports a top-down approach should be adopted (i.e. the design is approached primarily from the perspective of the conference application drivers such as media type, end-node profile, etc). An important aspect of modeling conferencing systems is to recognize the capabilities of emerging Internet protocols to decouple the signaling and media transfer functionality. It is possible to model at an even more granular level by differentiating between aspects of the signaling functionality (e.g. between registration, location and calling) and media transfer functionality (e.g. based on incoming and outgoing directions. Emerging Internet protocols such as SIP, SDP, RTP and RTCP offer some support in this regard but it may be necessary to design new protocols that can more effectively meet the requirements of more granular design models. While traditional conferencing has tended to use a completely centralized approach, future implementations will see a divergence in how signaling and media transfer are implemented. Conferencing in the NGN is likely to diverge on an even more granular basis. More work will be required to develop the modeling framework to reflect the multiplicity of permutations in the context of this granularity.

In the short term it seems that conference management (registration, security and location services) will maintain strong centralization characteristics while call signaling and media transfer will use a more distributed approach. The influence of routing protocol and related ideas associated with path determination in data networks, however, could see the signaling management become more distributed in the longer term. Although multicast delivery is ideal for scalable conferencing, the lack of widespread availability across the internet remains a limit to internet-based conferencing. In the interim, a hybrid unicast/multicast model will be best for media transmission and delivery with the central node acting as the root of the multicast tree for media delivery to participants.

SIP seems to be the preferred signaling protocol for NGN conferencing applications and is in a position to leverage existing IETF protocols such as RTP/RTCP while being extensible to add further functionality quickly. Although SIP can be implemented in a centralized and distributed fashion and using SDP can support calls with both unicast and multicast delivery of media. Java provides all the necessary API's for developing SIP telephony applications. In developing this system we have used the JAIN API's, the SIP reference implementation, Java's ability to transmit and receive multicast traffic, and a lot of the other standard java packages such as swing, net, awt, etc. The conference server architecture, using multicast rather than unicast

for the transmission of media, saves on both network bandwidth requirements and conference server processor requirements and allows for a more scalable system.

Looking Forward

One key area of research going forward will be to develop the framework for modeling multiparty conferencing systems to reflect the separation of signaling and media transfer and the improved granularity within each layer of functionality. Another will be to research emerging extensions to the SIP specification, such as the SIP SUBSCRIBE method and to identify potential for new extensions which would support more granular designs. A third focus of research will be to identify if an altogether new protocol with a more granular approach, (for example, one which considers application layer issues such as speaker/listener ratio, media type, etc), and which supports mobile ad hoc environments is needed for improved conferencing service creation.

References

Miladinovic, I. and Stadler J., *Multiparty Conference Signaling using the Session Initiation Protocol*

Schulzrinne, H., Handley, M.,Schooler, E. and Rosenberg, J. (1999),*SIP: Session Initiation Protocol*, IETF RFC 2543

Rosenberg, J. and Schulzrinne, H. (2001), *Models for Multi Party Conferencing in SIP*, IETF Internet Draft, work in progress

Postel, J. (1980), *User Datagram Protocol*, IETF RFC 768.

Postel, J. (1981), *Transmission Control Protocol*, IETF RFC 793.

International Engineering Consortium/Cisco, *Understanding Packet Voice Protocols*