A Scalable based Multicast Model for P2P Conferencing Applications

Mourad AMAD
University of Bejaia, Algeria
mourad_amad@yahoo.fr

Zahir HADDAD
Bejaia University, Algeria
haddad-z@hotmail.com

Lachemi KHENOUS
Bejaia University, Algeria
khenous_a@yahoo.fr

Kamal KABYL
LAMOS, Bejaia University, Algeria
k_kabyle2000@yahoo.fr

***Abstract*—Multicast conferencing is a rapidly-growing area of Internet use. Audio, video and other media such as shared whiteboard data can be distributed efficiently between groups of conference participants using multicast algorithms that minimize the amount of traffic sent over the network. This is far more effective than systems that maintain a separate link between each participant. On the other hand Peer-to-Peer (P2P) model is inherently characterized by high scalability, robustness and fault tolerance. With its decentralized and distributed architecture, a P2P network is somehow able to self organized dynamically. Peer-to-Peer model or architecture is well adapted to conferencing applications, effectively it can greatly benefit from P2P attributes such as: flexibility, scalability and robustness, particularly in critical environments such as: mobile networks. In this paper we propose a novel and scalable approach for P2P Conferencing. This model combines a call control and signalling protocol (SIP) with a "P2P" protocol (Chord) for maintaining (dynamically) a well stabilized and optimized architecture topology. This model is also based on an application layer multicast mechanism. Performance evaluation shows that our proposed approach benefits from SIP protocol flexibility, with the robustness and scalability of Chord protocol. The use of a multicast mechanism optimizes the overall traffic flow (control and media) and transmission efficiency.**

Key Words: Conferencing architecture, Application Layer Multicast, P2P.

I. Introduction

The N-way conferencing applications[4] connects few-to-few or many-to-many users, as opposed to streaming applications which provide one-to-many media distribution. Generally the conferencing groups (*audio or video*) are typically small, involving fewer than ten participants. Also membership usually changes dynamically and rapidly, where any member may join, leave or invite other participants to the conference at any time. This class of applications commonly requires significant processors and bandwidth resources, especially if multiple simultaneous users are allowed. For each group of N-participants, each one must encode its own media streams and transmits them to the $N-1$ other participants, while receiving and decoding media streams from the $N-1$ other participants. Thus processor and bandwidth resources can go over the limit and then constitute a critical issue. Existing conferencing architectures [16] using Session Initiation Protocol (SIP) or H.323 are generally based on a registration server for every domain. Scalability of such server-based systems, representing a single point of failure, is generally achieved with traditional methods such as DNS. P2P systems are distributed and scalable, thus providing a solution to the single point of failure problem [7]. Also conferencing applications deliver real-time information [4] between (*possibly large*) groups of people, requiring real-time communications and performance optimization.

Given this, our proposed conferencing model derived from an optimal combination of both protocols: Chord [1] (*for users/terminals discovery, maintaining topology stabilization*) and SIP [2] (*for call control and signalization*). This approach benefits from both protocol advantages, and from the use of an efficient multicast mechanism at application layer for data streams forwarding.
This paper is organized as follows: Section II gives a brief overview of the main architectures used for conferencing, with their limitations. In section III, we present the concept of application layer multicast. SIP protocol and Chord concepts proposed for "P2P" conferencing are resumed in section IV. Section V describes the concepts and the architecture of our proposed approach. We also give preliminary results related to performances of our architecture. Finally we conclude and give some perspectives and future works.

II. Conferencing architecture

Most of the existing SIP and H.323-based systems rely on a centralized architecture. In [7] a P2P model is proposed for user location purpose(*avoiding the use of a proxy server*). Conferencing applications require exchange of various types of media. H.323 Conferencing [9] is essentially based on a centralized server that uses a set of tightly integrated protocols to control sessions. SIP conferencing makes minimal assumptions about the underlying transport protocol; as it can be used with any type of transport protocol such as: UDP, TCP or TLS [15]. A comparison between SIP and H.323 [14], [6] shows that SIP provides a similar set of services to H.323, but with far lower complexity, rich extensibility, and better scalability, and facilities fixed mobile convergence. Also, due to its complete specifications and deployment, SIP can also be considered as one of the main "standard" protocol for IP signalling. For these reasons, SIP constitutes a good candidate for supporting signaling functions in our architecture. The two types of conference architecture (*Centralized and decentralized*) are described bellow.

A. Centralized architecture

The centralized approach requires a conference bridge or MCU (*Multipoint Control Unit*). All participants send audio, video, data and control streams to the MCU in a point-to-point mode (*see figure 1*). The central MCU controls and manages the conference. It also provides functions such as media mixing, switching or transcoding.
This centralized conference [10] provides main advantages such are: Conference control and management simplification, terminal/user capacity negotiation. Transcoding function can be provided by the central MCU which constitutes the focus for the signaling and media flow. This focus represents a single point of failure and requires expensive functionalities.

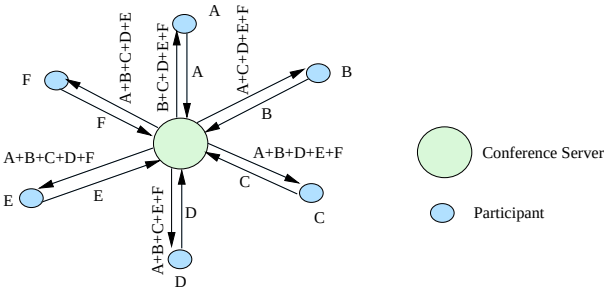

Fig. 1: Centralized architecture

B. Decentralized architecture

The second type of architecture (*decentralized architecture*) can be based on a full mesh topology connection [17] (*see figure 2*), or makes use of a multicast mechanism (*as our proposed model*).

The full mesh topology suffered from a lack of scalability, as each node must store $N-1$ information about other nodes (*for maintaining topology stabilization*). When the number of nodes increases, nodes with limited resources (*ex. mobile devices*) will present limited storage capacity for keeping this information. A decentralized architecture (*not full mesh*) is well adapted for this class of applications. Thus we propose for our model a Chord based architecture[1], and use an optimized algorithm for tree multicast construction. A multicast concept is presented in section below.

III. Application Layer Multicast

Application layer multicast aims to address scalability issues of unicast by distributing data replication process among the different group members, in an adaptative and efficient way. However ALM is not efficient as IP multicast in terms of data duplication. The nodes in ALM organize themselves into mesh or tree structures. More details are available on [3][5][11][12][13].
The next section gives a brief overview of Chord and SIP, as our contribution combines these two protocols for providing conferencing services.

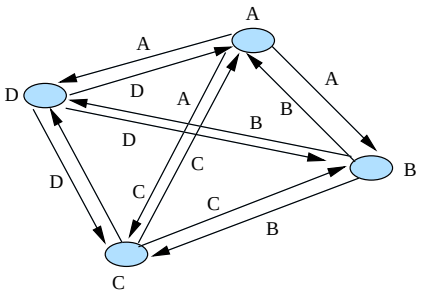

Fig. 2: Decentralized architecture

IV. SIP and Chord for P2P conferencing

Centralized and decentralized topologies (*full mesh*) lead to some critical issues such as scalability. The first one, requires high resources (*memory, bandwidth, processing*) particularly for the server, while the second one, also requires resources, high bandwidth for each participant. Our architecture is based on Chord protocol, it makes use of a decentralized architecture associated with a ring topology (*not full mesh*), and it uses SIP as signaling protocol.

A. SIP

Session Initiation Protocol (SIP) is completely specified in multiple IETF RFCs[2]. It is an IETF application layer protocol used for VoIP call control and signaling. SIP is a text-based protocol (*derived from HTTP*), used for establishment, modification and termination of all types of media sessions [2]. SIP defines two types of messages: requests and responses. The request type message is specified by its method. SIP RFCs defines multiple methods such as: INVITE, ACK, BYE, CANCEL, OPTIONS and REGISTER. Both requests and responses contain header and body that provide network or session description information. A SIP entity or service is identified by a unique name (SIP URL) using an E-mail type format address as: "server/user-name@domain/host-name".

B. Chord

Chord [1] is a P2P protocol which provides an efficient approach to resource location (*data or user*) issue. Chord uses routing queries to locate a key with a small number of hops $O(ln_2(n))$, even if the system contains a large number of nodes. It is characterized by its: simplicity, performance and robustness. It adapts efficiently as nodes join or leave the system dynamically. A node in Chord is identified by a unique identifier, this last is obtained from IP address, using distributed hash table (DHT) like SHA-1 or SHA-2. Each node in Chord maintains a routing table of (*at most*) m entries called the Finger table ($N = 2^m$ *where* N *is the nodes number in the system*). The i^{th} entry in the table at node n contains the identifier of the first node s that succeeds n by at least 2^{i-1} on the identifier circle. Chord is organized according a ring topology (*see figure 3*). Chord lookup protocol complexity is $O(ln_2(n))$.
The scalable approach proposed in this paper combines and

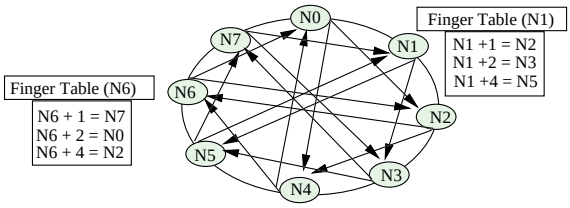

Fig. 3: Chord Architecture

takes advantages of both protocols described above: Scalability, robustness, stabilization algorithm, efficiency (*when a node joins or leaves the system*), interoperability, simplicity and security which characterized SIP protocol. So the main objective is to built a peer-to-peer decentralized conferencing topology based on Chord for assuring scalability and fault tolerance, while using the "standard" SIP for signaling. In one-to-many or many-to-many applications, each node has to broadcast "data" streams for M participants ($M > 1$). For optimizing data streams forwarding, we use an efficient application layer multicast mechanism build on top of the Chord layer. Section 5 presents and describes our contribution, particularly the architecture and protocol model for providing conferencing services.

V. Novel conferencing approach

The proposed approach combines both protocols described above: Chord for participant's discovery, localization, and topology stabilization management, and SIP protocol for signaling and call control.

A. Functional principle

When a node wants to join the conference group (*see figure 4*), it must find a SIP server addresses using DNS. It sends a SIP REGISTER message for authentication. Using bootstrapping mechanism[8], it joins the conference group. The second step for the new participant is to use one or more existing protocols such as ICE[18], for discovering and traversing NATs and Firewalls. The third step is to send a Join message to any group member. If the join process fails, the participant waits until a "time-out" period, then the process is repeated from the beginning. If the join process succeeds, the participant (*node*) takes a place in the system and initiates its finger table. The stabilization algorithm is executed for maintaining the correctness of the successors list along the ring. If a node wants to leave the conference, it sends a leave message to all nodes identified in its finger table, informing them to its leaving. The figure 4 resumes and describes this process.

In our approach, each node needs to know a limited topology related to the existing connections, for maintaining links correctness (*finger table*) along the Chord based ring. This topology is used to build a multicast tree for transmitting media stream to a set of receivers. The information about other nodes is stored in a specific data structure called Adjacency matrix. For data streams forwarding, a transmitting node needs

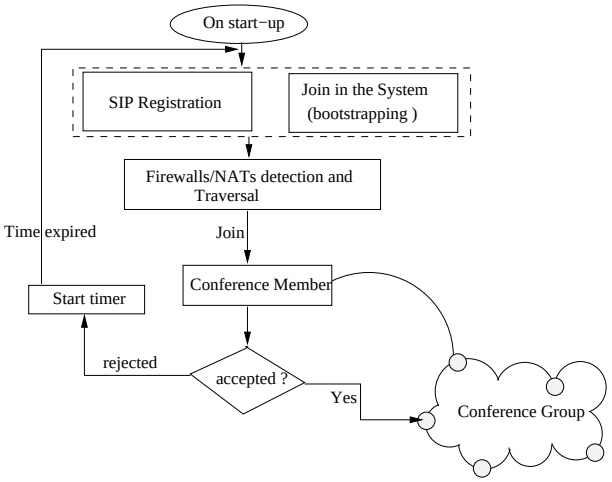

Fig. 4: Architecture-functional schema

to store IP addresses of receiving nodes. The next sub-section describes the multicast tree construction process.

B. Multicast tree construction

The Adjacency Matrix: Based on the finger table defined in Chord architecture, we define the adjacency matrix (denoted Adj_Mat) as:

Adj_Mat[i,j] = 1 : if there is a link between node N_i and node N_j (*link non bijective*), else it is equal to zero. The Adjacency matrix related to figure 3 is shown table I.

	N0	**N1**	**N2**	**N3**	**N4**	**N5**	**N6**	**N7**
N0	1	1	1	0	1	0	0	0
N1	0	1	1	1	0	1	0	0
N2	0	0	1	1	1	0	1	0
N3	0	0	0	1	1	1	0	1
N4	1	0	0	0	1	1	1	0
N5	0	1	0	0	0	1	1	1
N6	1	0	1	0	0	0	1	1
N7	1	1	0	1	0	0	0	1

TABLE I: The adjacency matrix (see figure 3)

Based on the adjacency matrix, we introduce an algorithm for multicast tree construction, with rapid convergence.

Based on this algorithm, we can build a tree with any node as

Algorithm 1 : Multicast tree construction algorithm

1: begin
2: if (N_i is a source node) **then**
3: Send message Child (N_i) to all nodes in its finger table
4: else N_i is not a source
5: At the reception of the messages Child (N_i) forwards this message to all nodes in its finger table excepts those in sets **A**, **B** or **C**
Where:
A: the set of nodes which precede N_i (*Parent*)
B: the set of nodes at the same level as N_i (*Brothers*)
C: the set of nodes which are child of its brothers and those last have an identifier numerically lower than that of N_i
6: End.

root, and connect it to all other nodes in the system. Figures 5.a and 5.b show the global multicast tree respectively with N_0 and N_3 as root.

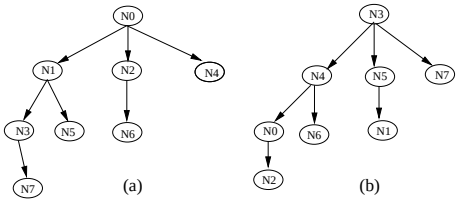

Fig. 5: Global Multicast tree

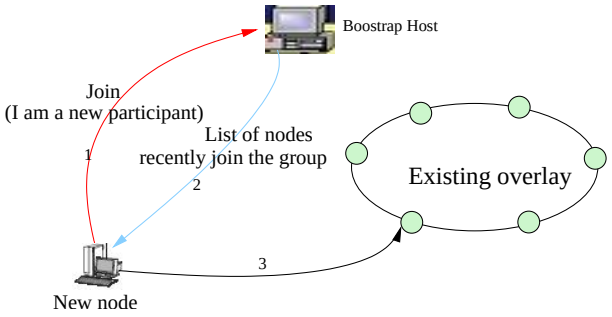

Fig. 6: The join process

C. Shared adjacency matrix

The Adjacency matrix can be viewed as a global knowledge base, associated to the list representing all the current participants. This data base is maintained in each node. So this matrix needs to be shared among all nodes. To limit and optimize the necessary resources (*ex. memory space*), which are critical in some environments (*ex. PDA, mobile terminal...*), we implement it as a matrix of bits. Then, for any change in finger table, the corresponding node updates this entry and sends it to all successors. The shared adjacency matrix is based on a distributed algorithm described below:

Algorithm 2 : Shared adjacency matrix

1: begin
2: For any changes in finger table (*neighboring*) or at reception of a new entry of adjacency matrix **do**
→ Update this entry.
→ Send this entry to all successors in the finger table.
3: end.

D. Join, leave and fault tolerance

In a dynamic environment, nodes can join or leave at any time. The main challenge is to preserve the ability to locate and update every key/user in the network. For this, "Bootstrapping" constitutes a vital core functionality, required by every Peer-to-Peer overlay network. Nodes intending to participate in such overlay network initially have to find at least one node that is already part of this network. Four solutions applicable for the Bootstrapping problem exist [8] and are resumed as : Static Overlay Nodes-Bootstrapping servers, Out-of-bande Address caches, Random Address Probing or Employing Network Layer Mechanism. To reduce system complexity, we advocate Static Overlay Nodes-Bootstrapping servers for our proposed model. The figure 6 describes this process.
When a node leaves the system, some nodes must update their finger table. After a failure detection at node n by its neighboring nodes, this last node invokes the join operation process for localizing one node. Then the Chord stabilization algorithm is executed.

VI. Analyze and Discussion

The analysis of the "overhead" (*control messages traffic*) generated by each participant in the system is represented[1] figure 7 and 8. For this we consider the three types of conferencing architecture (*centralized, decentralized full mesh and based multicast tree*). These figures show that the use of an application layer multicast mechanism, combined with a peer-to-peer architecture provides satisfactory results. As

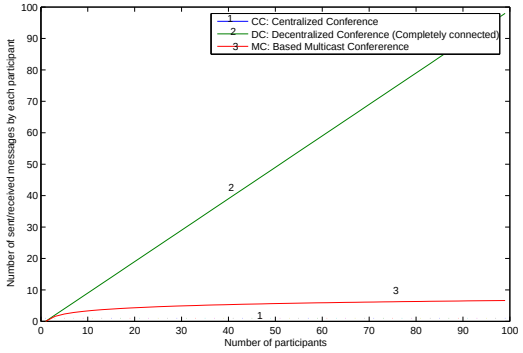

Fig. 7: Number of sent/received messages for centralized, decentralized full mesh and based multicast architectures.

shown in figures 8, for the centralized conference, the server needs to send $n-1$ messages for forwarding data flows to all participants in the system, and it receives a single data flow from each active participant (*at most* $n-1$ *nodes*). The other nodes send only one control message for forwarding its own data flow and receive one single message from the server. For the decentralized conference (*full mesh*), each node sends $n-1$ messages[2] to other participants and receives $n-1$ messages from them.

[1]Results are obtained from Matlab V7
[2]Message is a data flows

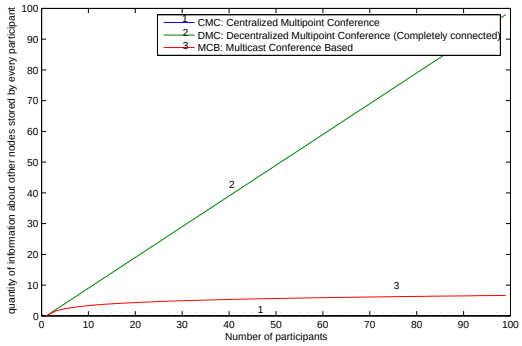

Fig. 8: Quantity of information stored by each participant

In based multicast decentralized conference architecture (*our approach*), each node sends $O(ln_2(n))$ messages to other participants and receives $O(ln_2(n))$ messages from them. Table II shows a numerical examples for $n = 32$, $n = 64$ and $n = 128$.

	CC	DC	MC
n=32	1	31	5
n=64	1	31	6
n=128	1	31	7

TABLE II: numerical examples

VII. Conclusion and Future Works

The N-way and multicast conferencing constitute a critical application for Internet. Audio, video and other type of media can be distributed efficiently between conference groups using multicast algorithms, minimizing the amount of traffic sent over the network. This is far more cost effective than maintaining a dedicated link between each participant. In this paper, we have proposed a based multicast architecture for conferencing. It is based on an efficient combination of P2P (*Chord*) and SIP protocol. The combination provides the main following characteristics : simplicity, robustness, scalability and fault tolerance. Then, we have defined an efficient application layer multicast algorithm for data flows forwarding. Interoperability is also facilitated by using SIP protocol but also, by the limited number of control messages generated by each peer. In term of future works, we envision to apply and combine a distributed QoS (*Quality of Service*) management, by extended our architecture to support a "lookup" mechanism (*for selecting a path or connection with the specified and requested QoS parameters*), and extended our model for other P2P networks. Finally a complete implementation will be achieved by using and extending existing P2P middlewares (ex. JXTA[3]).

References

[1] Ion Stoica, Robert Morris, David Karger, M.Frans Kaashoek and Hari Balakrishnan, *Chord: A Scalable Peer to peer Lookup Service for Internet Applications*, SIGCOMM'01 ACM, 2001.

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler, *SIP: Session Initiation Protocol*, RFC 3261, June 2002.

[3] S. Ratnasamy and M. Handley and S. Shenker, *Application-Level Multicast using Content Addressable Networks*, In Proc of Intl workshop on Networked Group Communication (NGC), November 2001.

[4] Peter T. Kirstein, Ian Brown and Edmund Whelan, *Secure Multicast Conferencing*, Technical report University College London.

[5] A. Rowstron and A-M. Kermarrec and M. Castro and P. Druschel, *Scribe: A large-scale and decentralized application-level multicast infrastructure*, IEEE Journal on selected Areas in communications, 20(8):1489-1499, October 2002.

[6] Jiann-Min Ho, Jia-Cheng Hu and Peter Steenkiste, *A Conference Gateway Supporting Interoperability between SIP and H.323*, ACM Multimedia 2001, P421-P430.

[7] Kundan Singh and Henning Schulzrinne, *Peer-to-Peer Internet Telephony using SIP*, NOSSDAV 2005 P63-P68.

[8] Curt Cramer and Kendy Kutzner and Thomas Fuhrmann, *Bootstrapping Locality-Aware P2P Networks*, 2003.

[9] O. Levin, *H.323 Uniform Resource Locator (URL) Scheme Registration*, RFC 3508.

[10] *http://www.ietf.org/html.charters/xcon-charter.html*, .

[11] D. Pendarakis and S. Shi and D. Verma and M. Waldvogel *ALMI: an Application Level Multicast Infrastructure*, In 3rd USENIX Symposium on Internet Technologies, San Francisco, CA, USA, Mars 2001.

[12] S. Ratnasamy and M. Handley and S. Shenker *Application-Level Multicast using Content Addressable Networks*, In Proc of Intl workshop on Networked Group Communication (NGC), November 2001.

[13] L. Mathy and R. Canonico and D. Hutchison *An Overlay Tree Building Control Protocol*, In Proc of Intl workshop on Networked Group Communication (NGC), Page 76-87, November 2001.

[14] Henning Schulzrinne and Jonathan Rosenberg, *A comparison of SIP and H.323 for Internet Telephony*, Network and Operating System Support for Digit Audio and Video (NOSSDAV) Cambridge, England, Jul. 1998.

[15] Henning Schulzirnne and Jonathan Rosenberg, *Signaling for Internet Telephony*, Technical Report CUCS-005-98, Columbia University, New York, Feb. 1998.

[16] Kundan Singh, Gautam Nair and Henning Schulzrinne, *Centralized conferencing using SIP*, In Internet Telephony Workshop, New York, Apr. 2001.

[17] Ling Chen, Chong Luo, Jiang Li and Chipeng Li, *Digiparty-A Decentralized multiparty video conferencing system*, Microsoft research Asia. 2003.

[18] Victor Paulsamy and Samir Chatterjee, *Network convergence and the NAT/Firewall problems*, proceedings of (HICSS'03), IEEE Internet computing, 2003.

[3]Project JXTA: "http://www.jxta.org/