Reliable and Scalable Groupcasting for P2P Replication-Based Collaborative Systems in Wireless Environment

(Vincent KWITONDA, Ryo NISHIDE, Hideyuki TAKADA)
Graduate school of Science and Engineering, Ritsumeikan University

Abstract - Collaborative applications are being revolutionized in schools, due to the proliferation of tasks to be accomplished by students. They are many collaborative activities which may be amenable to technological support. Groupware to support these collaborative works have raised research interest, in the context of CSCW. Profound advancements in P2P technologies have proven the capability to provide client-server functionality, at a much larger scale and much lower cost. CUBE, an object replicating framework for synchronous groupware in support of collaborative applications attempted to promote collaborative activities but it still has inherent multicast deficiencies, such as reliability and scalability. In this paper, we demonstrate the feasibility of new CUBE architecture and discover the performance profile in Wi-Fi environment and determine the impact on the applications running on it. A simulation session using NS3 ensures the fairness of bandwidth sharing between multicast flows in wireless environment.

Key words – Peer-to-Peer, Distributed systems, Protocols, Performance evaluation, JGroups, Workload, Wireless network, Packet delay

I. INTRODUCTION

Collaborative activities are gaining momentum in many sectors recently in educational sector, for instance in schools students accomplish a common goal collaboratively through real-time communication and interaction, each working to the benefit of all. This is due to the proliferation of the tasks the students have to accomplish and the volume of work in these modern times thanks to the technology and globalization. Different entities can design a learning community with shared goals [1][2]; they can organize collaborative process by developing shared script of joint activities in a classroom setting. A typical example is the usage of collaborative creation application with which students can utilize tile programming and 3D graphics. For instance, in its drawing space, they can create a 3D object with 3D blocks such as a sphere, a cone and a square by drag-and-drop operation, they can create these objects collaboratively and the operations for deployment are shared among separated computers synchronously [4].

Groupware to support these collaborative work have been researched on in the context of CSCW (Computer-supported cooperative work) in a distributed environment[10][11], users and applications interacting with a distributed system in a consistent way, regardless of when or where the interactions takes place, allowing real-time communication[14]; that’s how CUBE [4] (Collaborative Universal Basic Environment) framework came into being, to allow developers to conceive synchronous collaborative applications. However, the real-time interaction on synchronous applications has a much stricter network requirements in terms of update rate, message throughput and latency than asynchronous applications.

As these synchronous collaborative applications require technical and complicated factors such as
interprocess communication, synchronization, concurrency control and session management, it necessitates a framework for synchronous groupware to cover up these complicated factors. In this research, our goal is to work on the reliability of this framework CUBE, by tailoring JGroups[7], a Java communication toolkit, on CUBE lower layer tasked to assume multicast communication among the nodes and determine to what extent CUBE is scalable in Wireless environment (Wi-Fi) as students are entitled to move around the place to form groups; which signifies the necessity to run CUBE in a wireless environment. In order to guarantee strict QoS metrics, it is necessary to evaluate, improve and predict performance of the CUBE in a wireless environment (Wi-Fi) and the impact on the applications running on it with respect to different load conditions and variation of the number of nodes for the sake of management strategies.

In this paper, we present a new architecture of groupware CUBE which allows developers to conceive educational synchronous applications in real-time, in a P2P paradigm. In order to evaluate the effectiveness of the architecture, we make a performance evaluation in a Wi-Fi environment using NS3 simulator. The performance and scalability of CUBE are presented in the context of clustered peers. As a result, we ascertain the effectiveness of CUBE sharing multicast flows in a wireless environment.

The two main contributions of this work are to establish a reliable groupcasting and to specify the performance characteristics of CUBE applications in order to support real-time collaboration between different entities.

The rest of paper is organized as follows. Section 2 presents the social and technical background of synchronous groupwares in a P2P environment and their requirements. Section 3 presents the reliable service to facilitate a reliable multicast communication, and the protocol mechanisms used are presented with respect to the defined requirements of section 2. Section 4 describes the result, with an analytical scenario as well as a simulative evaluation of the overall CUBE framework. Section 5 concludes the paper.

BACKGROUND
Our work builds on previous work from collaborative technologies groupware, specifically CUBE framework [4]. In CSCW [10][11], researchers have considered several problems in real-world development and deployment of groupware, and have presented toolkits (e.g., [28][29]). However, we are still unaware of their performance tests.

Previous researchers noted that real-time groupware is different from other types of distributed systems in that it sends several types of messages with varying quality-of-service requirements [30]. There are a number of types of interaction which can be supported by groupware. Application using mouse movement, is the type of interaction CUBE supports. This can happen in a shared drawing application. Looking at the requirements of these applications, mouse-position messages are small, but mouse movement is very difficult to predict, and sometimes may require a high message rate with frequent message updates.

Message rate, message size and network delay are the QoS requirements that can be considered for groupware systems. Others are, the allowed latency and message ordering.
Reliability and Scalability Requirements in CUBE
CUBE conceptually uses objects mirroring as shown in Figure 1, where operations are replicated on all nodes identically. Reliability, among other things, includes lossless transmission of a message to all recipients, with retransmission of missing messages, ordering of messages; fragmentation of large messages into smaller ones and reassembly at the receiver’s side.

Object mirroring is for replicating and sharing data in real-time on replicated architecture. It has the following features:

- Mirroring of objects to replicate objects and keep their state identical in nodes for components of applications to be processed in parallel.
- Proxy to hide processes for the mirroring from developers, and provide a programming method as if they write source codes for a standalone application.
- Bounding of replication domains to define where objects are mirrored and reduce communication costs for mirroring [4].

CUBE realizes a replication of objects on the replicated architecture. The object mirroring on Java supports creating applications for synchronous collaboration. Objects Mirroring supplies objects called Mirror Objects as shared data and kept identical constantly. A term mirroring is defined as keeping an object state identical on each of the nodes [4]. To maintain the consistency of replicated objects, CUBE uses multicast communication. In terms of communication in collaborative systems, multicast seems the most natural way to perform group collaboration. However, unreliable nature of IP multicast is not sufficient to guarantee most of the desired functionalities.
A reliable multicast has to comply with the following features:

- **Integrity**: the integrity means that a correct process which is not faulty, delivers a message at most once. For instance, a process ‘P’ delivers a message ‘M’ at most once.

- **Validity**: the validity means that if a correct process sends a message, it has to eventually deliver the same message. For instance, if a correct process multicasts message ‘M’, then it will eventually deliver ‘M’ itself.

- **Agreement**: the agreement means that if a process delivers a message ‘M’, then all other correct processes in group(G) will eventually deliver ‘M’, which signifies a property of ‘All or nothing’ [24].

The current approach for communication used by CUBE is UDP multicast, with UDP protocol on its multicast socket layer. The reliability is reduced due to the increased probability of lost packets resulting from collisions or errors [20].

In the context of CUBE, reliability will be achieved as long as the objects can be mirrored by all group members, potentially both clients and servers with a lossless transmission to all recipients, with retransmission of missing messages. Object mirroring architecture is based on the replicated computing model. This architecture has advantage of a high fault-tolerance as the objects are kept identically in the same nodes where each group member is potentially both a client and a server.

The communication between processes on CUBE allows keeping the same state of objects identically in all of the nodes being in the collaboration.

![Figure 2. Object replication on CUBE](image)

To share objects among the nodes, replicas are kept consistently in all nodes on CUBE[4]. Those objects are consistent when the copies are always the same. Keeping the replicas consistent requires synchronization, which is inherently costly in terms of performance.

Next, the scalability of a system can be measured along three different dimensions, according to
A system can be scalable with respect to its size, meaning that we it can easily add more users and resources to the system.

A system can be scalable with respect to geographical scalability, users and resources lying far apart.

A system can be scalable administratively.

The scalability of CUBE is presented in the context of clustered peers.

**SYSTEM MODEL**

We consider a network consisting of a dynamic collection of peers that communicate through message exchanges. Each peer is uniquely identified by an ID (e.g. composed by IP address and port), required to communicate within the peers in a P2P environment [26]. The interaction between processes that constitute a P2P is equal, from a high-level perspective, whereby each process will act as a client and a server at the same time.

The P2P paradigm improves system performance by decentralizing the service load, it represents the concept of sharing resources, it dictates a fully distributed, cooperative network where nodes collectively form a system without any supervision; Profound advancements in P2P technologies have proven their capability to provide the traditional client-server systems at a much larger scale and lower cost. P2P networks are favored over centralized servers as they provide a higher robustness with no single point of failure; peers’ collaboration can run on multicast approach (IP, Overlay multicast and Application layer multicast). With multicast, it means transmitting the same data concurrently to multiple receivers, instead of sending a separate copy of the data to each individual receiver.

**Architecture principles - Groupcasting on CUBE**

Distributed computing systems have to be organized logically and physically. CUBE middleware forms a layer between a distributed platform and applications on top of it as shown in Figure 3. The main actors in the architecture of CUBE are JVM, CUBE Framework and upper layer allowing developers to conceive synchronous collaborative applications.

![Figure 3. Architecture of CUBE](image)

As illustrated in Figure 3, the current architecture is using UDP multicast [23] inappropriate for reliable message delivery in CUBE framework.
UDP is scalable but not reliable whereas on CUBE, messages drawn by the applications running on it have to be delivered according to a certain order. To address the deficiencies caused by this lower layer, a new paradigm is expounded in the section below.

**Groupcasting based on JGroups and Programming Model**

In an effort to fill the void left by the lower network layer on CUBE which doesn’t allow a smooth collaboration between peers, we have extended an existing toolkit for reliable communication; JGroups[7], chosen as the most suitable platform for reliable communication to replace the lower level of CUBE. One of its key features is its flexible protocol stack, which can be configured and extended depending on the communication needs. The protocol stack has different functionalities such as reliability, state transfer and etc. A stack is a sequence of protocol names and their optional initialization parameters.

As illustrated in Figure 4, CUBE model is extended with JGroups[7], a reliable multicast toolkit as it can support sequence ordering, failure detection and flow control and encryption of messages.

![Figure 4. Architecture of new CUBE framework](image)

Processes can join a group, send messages to all members or single members and receive messages from members in the group. The system keeps track of the members in every group, and notifies group members when a new member joins, or an existing member leaves or crashes. A group is identified by its name. When a process joins a non-exiting group, that group will be created automatically. A member can be part of multiple groups.

Groupcasting on JGroups is based on the fact that to join a group and send messages, a process has to create a channel and connect to it, using the group name, wherein all channels with the same name form a group. JGroups is a commonly used open-source toolkit for group communication among other communication toolkits [28][29]. Different protocols of JGroups stack exchange event up and down the stack in order to share information. As can be seen on Figure 4, JGroups is used as the lowest level communication API over IP for multicast communication in the group. Each JGroups channel (JChannel) is initialized with a protocol stack specified in a string or an XML file.

The communication on CUBE is based on this low-level message passing offered by the underlying network. The communication is divided up into level or layers, each layer dealing with specific aspect of communication. As it
can be seen on Figure 5, JGroups is used as the lowest level communication API over IP for multicast communication in the group.

Group management services support multiple group creation. Groups are essential for collaboration in CUBE, since all actions are performed inside the scope of a group. The group membership service maintains a list of active nodes. It handles the requests to join and leave the cluster. Groups are created by providing the group name and then, multicast address and port mapping are provided from the underlying services. Each member can join several groups at the same time and membership on each group is provided as a list of members that currently belong to the group, together with notification about member join/departure events.

Message delivery:
The message delivery is provided with unicast reliability or with multicast in addition to retransmission of missing or dropped messages towards the full delivery ratio of the message transmitted. The mechanism of message delivery is based on ‘FIFO ordering’ with a sequence or token-based. The processes maintain a group specific sequence number.

Deployment view:
Cooperating entities are able to join or leave the application freely and they can easily grow to a classroom number, in our case, we took 30 students each using one node as shown in Figure 6.
The wireless-enabled peers have to be within the range of base station. The wireless clients generate the data packets and send them to the other wireless clients via multicasts.

EVALUATION AND SIMULATION RESULT

To find out how CUBE may perform better in a classroom setting of 30 peers in wireless environment, the key dimensions on which the quality is evaluated is the overhead generated by messages retransmission. To know how many peers can scale much better with a flow control management in a Wi-Fi environment, a performance evaluation is performed. Furthermore, to show the behavior of the framework with different parameters such as an increase or decrease of peers, we do a simulation.

We have used a network simulator NS-3.13 to implement the multicast collaboration. In this section we will describe the simulation results. The following metrics are used for the performance evaluation using the simulator: (1) Average time of packet delay. It is defined as the time period from the start of an multicast source sending a certain multicast frame to the end of the successful reception by all receivers.

A. Simulation Setting
We have set up the simulation with 30 nodes. Some of the simulation parameters are shown in the Table 1.
Table 1: Simulation Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Description</td>
<td>CUBE</td>
</tr>
<tr>
<td>Cluster size</td>
<td>30</td>
</tr>
<tr>
<td>Number of senders</td>
<td>Every node</td>
</tr>
<tr>
<td>Message size</td>
<td>1 K</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>54 Mbps</td>
</tr>
<tr>
<td>JVMs</td>
<td>SUN JDK 6</td>
</tr>
<tr>
<td>Transport Protocol</td>
<td>IP multicast –based (UDP)</td>
</tr>
</tbody>
</table>

B. Simulation Result

We carried out assessments testing retransmission rates with a standard 1-kilobyte equivalent to 5 clicks by a user. We inspected packets sent with retransmission to determine the average retransmission time and rate. Our tests are shown within 30 nodes sharing 54Mbps of bandwidth as shown in Figure 7.

Figure 7. Average overhead between 30 nodes

As shown in Figure 7, the x axis shows the number of nodes and the Y axis shows the overhead generated by message retransmission. As for the lines in the plot area, 1/6 represents the percentage of nodes having packets dropped and need to be retransmitted, 1/3 represent one third of 30 nodes dropping data in a need to be resent, 1/2 represent a half of failed peers whose packets need to be retransmitted and finally 1/1 shows the whole failure of all 30 nodes.
CONCLUSION

In this work, in the light of what we have seen above, we conceive a reliable CUBE framework tailoring JGroups on its multicast socket layer for smooth collaboration between peers involved and determine how big CUBE is scalable in wireless environment (Wi-Fi). Having presented the evaluation results in section above, we conclude that the CUBE architecture allows a good platform for multicast collaborative applications based on the simulations results.

REFERENCES


