Increasing the Trustworthiness of Collaborative Applications

I

Mamdouh Babi
Department of Electrical and Computer Engineering, Cleveland State University, USA

Wenbing Zhao
Department of Electrical and Computer Engineering, Cleveland State University, USA

INTRODUCTION

Computer Supported Collaborative Work (CSCW) refers to the interaction of two or more people to work collaboratively on the same application (Dillenbourg, 1999). The history of CSCW can be traced back to the late 1960s, when many visionaries like Ted Nelson, Douglas Engelbart, and Alan Kay saw the potential of digital media to be the primary method of communication amongst people. In the early 80's with the development of personal computers, modes, and other preferable devices, CSCW start to take a new trend in society. In 1998, Irene Greif published a book titled "Computer-Supported Cooperative Work: A Book of Reading" (Greif, 1988). The book explained the motive and the inspiration behind Douglas Engelbart's work in CSCW (between 1963 and 1984). Although Engelbart is well known for his invention of the computer' mouse, he has much broader vision of human interaction using the digital media. After Engelbart's invention, users start to collaborate online using small suite of tools like LISTSERV, Netnews, MUD, and other developed tools. In the 90's, with the development of branded technology, real time applications such as video conferencing, texting, and instant messaging start to take place in our society. Immediately afterward, more advanced real time applications, such as whiteboard, meeting facilities, and applications sharing, were put to use allowing users to interact in real time environment. Today, CSCW is more popular than ever. CSCW has become the trend of sharing documents and working collaboratively on the same task from different locations. The locations can be in the same geographical location or can be across the globe.

In this article, we will focus on how we can increase the trustworthiness of collaborative applications. Our work deals with the integration of BFT and Operation Transformation (OT) and the insurance of Byzantine fault tolerance (BFT) logging and undo/redo capability so the damages caused by malicious users can be reversed, returning to exactable state. We will use Collaborative Editing as a running example in our discussions.

BACKGROUND

Collaborative work applications involve shared views by multiple users. For example, in Collaborative Editing, multiple users can view, edit, and save the same document at the same instant. Therefore, any infrastructure in collaborative work must support consistency and some type of concurrency. Some systems support strict consistency mode. In this type of architecture, all participants will share the exact same view. The mechanisms in these types of setting are supported by floor control and pessimistic locking. Other systems support replicas divergence. The mechanisms of these types of setting are supported by optimistic locking and operation transformation strategies. The developer needs to make a clear choice of the type of consistency control that he/she should be using by applications. The choice made represents the form of consistency against performance and scalability. No matter what choice is made, any system designed should contain the dependability factor. For a system to be dependable, it must be reliable, available, safe, secure, and trustworthy. A solution for protecting real-time collaborative editing systems against Byzantine faults was presented in a recent paper (Zhao & Babi, 2013). A key contribution of the proposed solution is to maximize the protection of a collaborative editing system without resorting to

DOI: 10.4018/978-1-4666-5888-2.ch424

the use of additional redundant resources such as a separate Byzantine fault tolerance service cluster. Next, we will define quantitative dependability measures:

- **Reliability:** The measure of continuous delivery of proper service or equivalently of the time of failure. In other words, it is the probability of the system to survive, despite failure, over an interval of time.
- **Availability:** The measure of delivery of the correct service at any given time. Meaning, the system must be available to deliver the service at all time.
- **Safety:** The acceptable misshape risk that does not have any type catastrophic consequences, within all constraints of operation throughout all phases of the systems lifecycle.
- **Secrecy:** The ability of the system to preserve confidentiality of all information and data at all time.
- **Trustworthiness:** The ability of the system to do what it exactly expected to do despite all type of disruption including: environmental disruption, users, operator errors, and all types of attacks, malicious and non-malicious, by hostile parties.

INCREASING TRUSTWORTHINESS OF CSCW APPLICATIONS

Threats and Control in CSCW

We have seen an ever-increasing demand for Computer Supported Collaborative Work (CSCW) applications. A CSCW application might be subject to various hardware and software faults. One of the most dangers threats to CSCW applications are malicious attacks on them (often modeled as Byzantine faults). More precisely, a Byzantine Fault refers to an arbitrary fault caused by accidental or malicious activities by an intruder who tries to cause damages to the system. A Byzantine Fault Tolerance (BFT) system is one that is able to tolerate Byzantine faults. BFT can be achieved by replicating the server in addition to the insurance that all server replicas can reach an agreement on each and every input. Such an agreement is referred to as a Byzantine Agreement (Lamport et al., 1982).

In addition to the threats to CSCW outlined above, and because we have multiple user using the same application and maybe the same page of application or even the same task, we need some type concurrency control to ensure that the changes to the shared state are consistent and conform to the users' intent while allowing multiple users to work concurrently. In Collaborative Editing, for example, participants working on the same application or document can see the expected state without any supervises. Operation Transformation is one of the mechanisms used to ensure consistency. In the next section, we will discuss Operation Transformation, and how it is used in CSCW. Then afterword, we will discuss undo/redo and how it can be used in CSCW.

Operation Transformation

Operation Transformation (OT) was originally used for consistency and concurrency control in collaborative editing applications that use plain text documents ("Operational transformation," 2012). The main idea of using OT is to achieve both convergence and intention preservation without imposing any type of restrictions on the user. The user can edit any object at any time without restriction. OT framework is based on the following:

- OT Control Algorithms: Determines which operation is transformed against other according to the concurrency or context relations.
- OT Transformation Functions: Determines how to transform a pair of operations according to types, position and other parameters.

Let us illustrate OT with the following example using simple text editing scenario. We will use two users 'Alice" and "Bob" for simplicity reasons. Let us assume that Alice and Bob are both working on the same document (Figure 1). Let us say that Alice want to insert the letter "x" at the letter string "abcd." Let us assume that Bob wants to delete the letter "b" from the same string "abcd" at the same instant of time.

Alice will request a command execution: