Distributed Command Framework for Online Tutorial System

Sandi Su Hlaing, Ei Chaw Htoon
University Of Computer Studies, Yangon
mietchawsu@gmail.com

Abstract

Distributed Command Pattern is a proposed pattern for solving the extra architectural design issues which need to be considered for connected systems than designing regular applications. The goal of this pattern is to "Have the same design for both standalone and connected systems". This pattern allows developers to concentrate on designing a regular application using the Command Pattern without ever thinking that this application is going to connect with other systems. By using the framework, developers can minimize the amount of required for communications between online tutorial server and clients since this system is built to send and receive any command easily by using distributed command pattern. In this paper, distributed command framework for Online Tutorial System is implemented so as to provide a good framework between Tutorial Server and its Clients.

Keywords: Distributed System, Command Pattern, Distributed Command Pattern, Client-Server Model

1. Introduction

Distributed Command Pattern is a good pattern when developers want to design a good system for connected systems. Designers need not worry about how to send necessary data to the server when some command gets executed and how to process received data from the server and then reflect on the UI of the connected applications. When a command gets executed locally on one running application, distributed design pattern takes care of the hurdles of executing the same command in all connected systems simultaneously in order to keep them in the same state. From a designer's point of view, the architecture remains as simple as a regular desktop application designed using regular command pattern having no network awareness, yet it performs all the necessary communications in order to execute the commands on all connected applications as if the command has been invoked internally.

This system is an online tutorial system and uses distributed command pattern theory to set up the connection between online tutorial server and clients. Furthermore, this system is built to overcome the problem of distributed systems whenever developers meet while setting up the connections between server and clients with the use of socket library. By using the nature of command pattern and bidirectional http channels, it can be said that there is a good connection between server and clients.

In this paper, online tutorial system is introduced by using distributed command framework. Section 2 in this paper depicts related works of this system. Section 3 of this paper explains distributed command framework architecture. In section 4, proposed online tutorial system architecture is presented in detail and conclusion is described in Section 5.

2. Related Work

Omar Al Zabir [4] constructs the distributed command pattern by using Command pattern from Design patterns [2]. Distributed Command Pattern is a pattern for connected systems which implements command pattern. It frees designers from thinking about the communication and helps them concentrate on implementing commands as if it is a regular desktop application. The framework takes care of the communication. It is implemented on Chat Application and explains how to build a new framework for connected system using Command pattern.

ParasadPerera [5] highlights the usage of command pattern in organizing computing tasks and managing them. Besides, this application describes the usage of command pattern in order to manage a queue of different types of jobs to be executed by several threads in a multi-threaded environment and also shows a basic scenario with the sample application how this can be achieved using the command pattern.

3. Distributed Command Framework Architecture

A distributed system is a system consisting of a collection of autonomous machines connected by communication networks and equipped with software systems designed to produce an integrated and consistent computing environment. Distributed systems enable people to cooperate and coordinate their activities more effectively and efficiently. The key purposes of the distributed systems can be represented by: resource

sharing, openness, concurrency, scalability, fault-tolerance and transparency. [1, 3]

There are several problems in distributed system. While designing the architecture of connected systems, the design needs to consider how commands are executed not only in the application where it originated, but also on all other connected systems. For example, in a chat application which is connected to a chat room, if one user writes a message, the message needs to be shown on the user's screen and also simultaneously on screens of all other users who are present on the chat room. This raises an architectural design issue where a command needs to be invoked on the originating application and necessary data be sent to the server in order to broadcast the message to all other connected users. Each connected user needs to receive the data from server, translate it, and then show it on the screen. [4]

3.1 Command Pattern

Command Pattern (shown in Figure 1) is a very widely used design pattern for both web and desktop applications. It allows developers to think and design architecture in terms of actions performed by the user and the system. For example, when a file is opened, a FileOpenCommand is issued and necessary code is written only for that command. This simplifies architecture design because when implementing an action, the job needed to do is narrow and focused. Command Pattern also forces design of a very reusable system because the small command classes are reusable by design and can be invoked either by the user or by any system. For example, a user can issue FileOpenCommand by selecting "Open" from "File" menu, or it can be automatically invoked by a plug-in or by a macro, or by another command which needs to open a file. As a result, the command is written only once and maintained in only one place, but it offers wide reusability. [2]

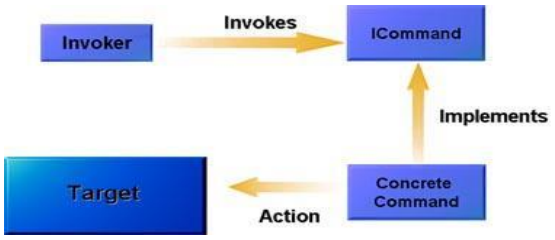

Figure 1. Command Pattern

3.2 Command Bus

In Distributed Command Pattern (DCP), Command Bus (depicted in Figure 2) is created following the idea of Message Bus pattern.

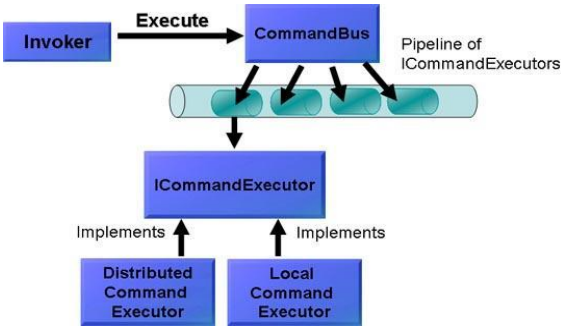

Figure 2. Command Bus

Message Bus introduces a bus which knows how to format a message and how to send the message via some channel. In DCP, Command Bus is also like a Message Bus. It maintains a pipeline of command executors which actually perform the execution of the command. Whenever a command is sent to the Command Bus, it informs all executors in the pipeline to execute the command. The executors work as channels and actually do the execution of the command. [6, 7]

Executors provide the execution environment for commands. Although the commands contain necessary code to be executed, command executors provide necessary support to the commands during execution. For example, imagine a Windows form which has a text box. A command needs to add some text to the text box when executed. But the command is a separate class and it has no reference of the text box on the form. So, the form, being a command executor, intercepts the command from the command bus, and provides the command object with the reference to the text box. It then executes the command.

One type of executor is local executor, where the command is executed in-process. Another type of command executor is called "DistributedCommandExecutor" which has the responsibility to send or receive commands via transporters. This type of executors are responsible for keeping all connected applications in-sync and provide the command distribution service.

However, the distributed executor does not execute a command directly. It just sends the command using a transporter to another connected application. On the other end, there is another distributed command executor which receives the command from its own transporter. It then notifies the command bus that a command is received and the bus again sends the command through the pipeline for execution. This time, the local executor on the other end receives the command and executes the command in its own context. This results in execution of the same command on another computer just as if the command was created and executed there. [4]

3.3 Distributed Command Pattern

Distributed command pattern solves the problems by introducing an architecture where the same command is invoked automatically on connected systems when any one system executes a command. This pattern offers a way to invoke a command in-process on a connected system from another connected system. There is no need to send a custom message containing required data to the server to notify about the command that is executed, and there is also no need for clients to read and parse a message in order to extract necessary data and then act according to the message. The command that is executed on a system is serialized and transmitted to other systems and then executed locally in order to simulate the same situation that occurred on the originating system. Pipeline nature and high level architecture of distributed command pattern are shown in Figure 3 and Figure 4. [4]

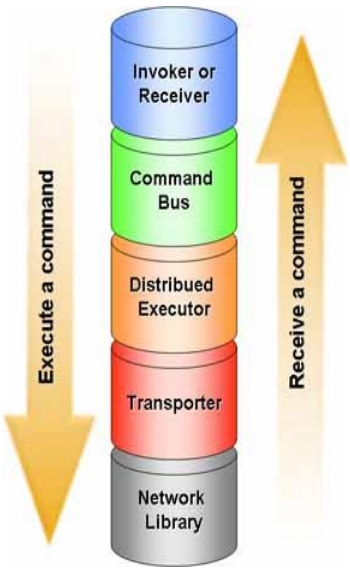

Figure 3. Pipeline Nature of Distributed Command Pattern

Transporter provides the communication service. It sits between a distributed command executor and the network communication library to provide transparent message transportation service. A transporter is created by the application according to the communication media it uses, and the application itself attaches the transporter with a distributed command executor. For example, when a client application connects to a server using a TCP/IP connection, it uses a transporter which has the ability to send and receive data using Sockets in binary mode. After successfully establishing a connection, a distributed executor is created for the transporter and the executor is put in the command bus for transmitting commands. [4]

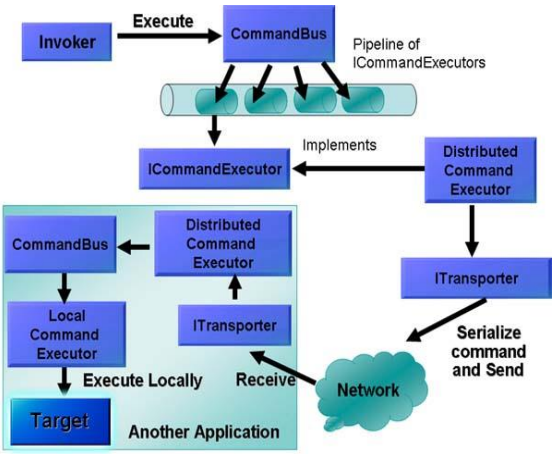

Figure 4. High Level Architecture of Distributed Command Pattern

When a command is sent to the command bus, the following execution occurs:

- Command bus passes the command to each executor in the command pipeline.
- If the executor is a local command executor, it executes the command directly.
- If the executor is a distributed command executor, it decorates the command using a RemoteCommand class (Decorator Pattern [2]). This class contains the actual command and additionally contains a transporter ID which identifies from where the command is being sent.
- The remote command is then sent to a transporter in order to send the command to a destination using some communication protocol.
- On the other end, the transporter layer receives the message. It creates the command object from the message.
- The command is sent to the distributed command executor which is attached with the transporter. The executor receives the command, and then it modifies the "Source" property of the RemoteCommand to set the transporter ID from which it received the command.
- The executor notifies its command bus about the arrival of the command.
- That command bus notifies all of its command executors in its pipeline to process the command.
- During this time, the executor which originally notified the command bus about the received command again gets called by the command bus as it is in its pipeline. But it sees from the source property of the remote command that, it is that executor which sent the command to the bus. So, it

does not again send the command to the transporter and thus prevents an infinite loop. [4]

4. Proposed Online Tutorial System

This system is implemented by using Distributed Command Pattern. The architecture of the system is shown in Figure 5.
In the system, there are two types of user.

- Exam server owner (Admin User)
- Exam client

The admin user can add new users, new questions, update questions, and delete questions. Besides, the admin user can load the questions according to the exam type and send them to clients to sit for the exam, and then, calculate marks after receiving all answers from clients and send them back to respective users.

The exam clients will login in the exam server and answer all questions given by the exam center. Moreover, the exam clients can cease the exam as soon as all questions has been answered and see the marks sent by server after calculating marks.

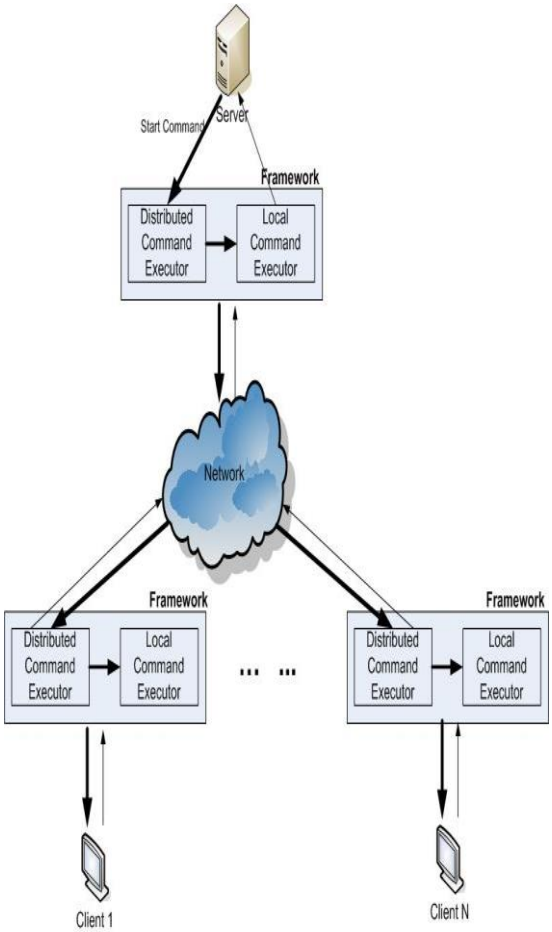

Figure 5. The Architecture of the System

The Exam Server has the following services for new user registration and new questions entry or old questions update or delete.

- New User Registration
- Questions Entry
- Questions Update
- Questions Delete

In the system, the client must login the server first to start the exam. Then, the 'Ready' has to be sent to the server so as to inform that the client machine is ready to sit for the exam. After questions have been sent from the server, the client needs to click 'View Question' to start the exam. While the client is sitting for the exam, the exam client can stop the exam and send all answers to exam server as soon as all given questions have been answered before the timer is expired. If not, all answers will be sent automatically to the exam server after the timer expired.

Looking at the Exam Server Site in detail, to start the server connection or to make successful connections among Exam Server and Clients, the Exam Server has the following commands:

1. Start
2. Load Question
3. Check All Clients
4. Send Question
5. Calculate Marks
6. Send Mark
7. End

During the processing time of commands, the Start Command , the Load Question Command, the Send Question Command, the Send Mark Command and the End Command call the following classes: 'Invoker' **,** **'**Command Bus', **'** ICommand Executor', **'**Distributed Command Executor', **'** ITransporter' whereas the Check All Clients and Calculate Marks call 'Remote Command', 'ITransporter', 'Distributed Command Executor', 'Command Bus', 'Local Command Executor', 'Receiver'.

For the Client Site, the Exam Users have the following commands to succeed the connections between Exam Server and Clients.

1. Exam Center Login
2. Ready
3. View Question
4. Send Answer
5. View Marks

While commands are processing, the Exam Center Login command, the Ready command and Send Answer command call 'Invoker' **,** **'**Command Bus', **'** ICommand Executor', **'**Distributed Command Executor' and **'** ITransporter', the View Question and the View Marks command call 'Remote Command', 'ITransporter', 'Distributed Command Executor', 'Command Bus', 'Local Command Executor' and 'Receiver'.

By using the Distributed Command Framework, the online tutorial system has the following benefits:

- A Command class is developed exactly the same way as it is developed for simple standalone systems which have no network awareness.
- There is no need for designers to develop custom protocols or design messages in order to send or receive necessary data for each command.
- When a command is invoked, it not only gets executed on the same process, but also on other connected processes and connected computers over the network simultaneously.
- It enables connected systems to be "in-sync" with all others without knowing about others' presence.
- It frees developers from thinking and implementing how to have all connected systems in-sync with each other as the exact same command is executed in all processes just as if the command was invoked locally in each connected system.
- Developers need not worry whether the command is invoked from the same process or is coming from another process.
- It reduces the amount of code needed for communication as there is a unified way to send or receive any command.

5. Conclusion

This paper is implemented the distributed command framework that users can use in connected system. By using this framework between Tutorial Servers and Clients, designers can reduce in developing the custom protocols or design messages in order to send or receive necessary data for each command. Moreover, it frees developers from thinking and implementing how to have all connected systems in-sync with each other as the exact same command is executed in all processes just as if the command was invoked locally in each connected system. Besides, it reduces the amount of code needed for communication as there is a unified way to send or receive any command. Finally, best connection system will be got and time can be saved.

References

[1] Comer, Douglas E, "*Internetworking with TCP/IP, Volume 1***"**, Principles, protocols, and architectures, Fourth Edition, ISBN 0-13-019353-4, 2000, 1995 by Prentice Hall Inc, 1995.

[2] Gamma, Erich et.al , " Design Pattern" , Addision-Wesley Professional; First Edition, 1995.

[3] George Coulouris, Jean Dollimore, Tim Kindberg, "*Distributed Systems CONCEPTS AND DESIGN***"**, Third Edition, ADDISON WESLEY, An imprint of Pearson Education, 2001.

[4] Omar Al Zabir, "Distributed Command Pattern – an extension of command pattern for connected systems", Code Project, 2005. [http://www.oazabir.com]

[5] ParasadPerera, "Use Command Pattern for Handling Queued Jobs in a Multi-threaded Environment", Codeproject., 2008. [http://www.codeproject.com]

[6] Snell, James M, *"Understand and implement the message bus pattern*" , IBM, 2004.

[7] Szymanski, Wytek, "Bi-directional HTTP Connection", Code Project, 2004.