Journal of Network and Computer Applications 34 (2011) 232–251

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Universal connection architecture for interactive applications to achieve distributed computing

Tzu-Chi Huang *

Department of Electronic Engineering, Lunghwa University of Science and Technology, Taiwan

ARTICLE INFO

Article history:
Received 18 February 2010
Received in revised form
14 July 2010
Accepted 14 July 2010

Keywords:
Interactive application
Distributed computing
Universal connection architecture (UCA)

ABSTRACT

While an interactive application (IA) occupies a certain degree of market and gradually becomes popular in our lives, e.g., database applications, management applications, and control applications, IA developers are faced to deal with many works not belonging to what the IA is supposed to have in its original design due to the emergence of distributed computing technology. IA developers can use the universal connection architecture (UCA) proposed in this paper to free them from the burdens of achieving distributed computing in the IA. IA developers can use the proposed UCA to focus on the design of the IA without learning other unrelated knowledge and the use of network APIs. IA developers can use the proposed UCA to connect the IA's components distributed over networks without changing the programming style or source codes of the IA. IA developers can use the proposed UCA to repeatedly evaluate and dynamically extend network functions of the IA with various modules at run time without recompiling the IA or linking the IA to a different module. IA developers can use the proposed UCA to utilize the distributed components written in different languages. With the proposed UCA, IA developers can immediately achieve distributed computing in the IA once core functions of the IA are finished. In this paper, IA developers can understand the proposed UCA, its overhead, and its performance.

1. Introduction

Among various available applications nowadays, an interactive application (IA) occupies a certain degree of market and gradually becomes popular in our lives. The IA includes database applications, management applications, control applications, etc. The IA works based on a query and response model, i.e. accepting commands from a user, processing the commands, and replying to the commands with results. The IA may accept text commands directly or those wrapped indirectly by a graphical user interface (GUI). When processing the commands, the IA may search a database for an entry of record, works like an agent to manage devices on behalf of users, or asks the underlying system to do a task. Finally, the IA replies to the commands with information or a brief message according to the processing result of the command.

Due to the emergence of distributed computing technology, however, the IA developers are faced to deal with many works not belonging to what the IA is supposed to have in its original design. The IA developers have to learn network knowledge, understand network programming, and make the IA capable of networking, e.g., providing users with network accesses or using networks to connect the distributed components. The IA developers have to evaluate, select, and further figure out the appropriate connection mechanism or topology to make the IA work well in the network environment. Besides, the IA developers may need to revise the programming style of the IA, e.g. supporting multiple threads, in order to handle numerous requests from networks. Furthermore, the IA developers have to learn how to partition the IA into different components and distribute them over networks. Finally, the IA developers may find that the product is not acceptable and needs to go through another aforementioned procedure. For achieving distributed computing, the IA developers nowadays cannot rest just after the IA is finished, but have to revise the IA repeatedly.

For achieving distributed computing in the IA, the IA developers have to suffer many negative consequences. First, they pay much time for learning other knowledge unrelated to the original IA design and prolong the application development time. For example, they need to learn network knowledge, know distributed computing concept, understand the meaning and usage of network application programming interface (API) in an operating system, etc. Second, they change the programming style of the IA for networking, which makes difficulty in maintaining source codes of the IA in the future. For example, they modify source codes of the IA for handling numerous requests with multiple threads, for connecting components distributed over networks, or

* Tel.: +886 2 82093211x5633, +886 920981224.
E-mail addresses: tzuchi.phd@gmail.com, tzuchi@mail.lhu.edu.tw

1084-8045/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2010.07.013