The Design of Real-time Transmission

Yun-Xiang WEN1,a,*, Wei-Jian CI2,b

1Class 20, Senior Grade 2014, The Wenjiang Middle School, Chengdu, Sichuan, China
2School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China

a2806393124@qq.com, b2036336710@qq.com
*Corresponding author

Keywords: Publish-Subscribe, Real-Time, Schedule.

Abstract. Some network services, such as Vehicle tele-operations, remote monitor requires time critical services. The publish-subscribe architecture has been designed to provide message transmission task. The paper presents a publish-subscribe communication model with QoS parameter indicator. The fair scheduling strategy guarantees the real-time service. The implementation of real-time publish-subscribe communication protocol is built on the top of standard UDP stack. It offers practical advantages for applications with time-critical data flows, which can provide control of timing requirement and reliability.

Introduction

Vehicle tele-operations, services for data collection from a sensor network, and services for system supervision with aggregation of information are required to deliver data in real-time. Such real-time applications have two main requirements for data delivery: time determinism and performance. Though the best service has been a popular access mode, its delay is not deterministic due to its access control. In this case, the publish-subscribe (PS) architecture is designed to provide critical time requirement transmission [1]. Events are sent to the subscribers once after particular event occur, thus the subscriber can get the data in real-time. In addition, PS systems free data publisher from waiting an acknowledgement of subscriber. Publisher can quickly deliver the data to receivers within deterministic time without any synchronous operations, and only one event is required to deliver to many subscribers.

This paper is organized as follows: Section 2 provides a motivation and related work for this work. Section 3 presents the PS model. Section 4 describes the design and implementation of PS model in detail. Section 5 presents some conclusions and suggestions for further development of this work.

Related Work

Currently there are multiple innovative approaches developed to provide real-time behavior in wireless-supported applications. The master-slave paradigm is the simplest communication mechanism, where the sender just needs to know the address of the receiver to send its messages. However, this approach is completely centralized and it does not offer multicast/broad-cast data exchanges. Existing publish/subscribe systems include: CORBA Notification Service [3], JMS [4], TIB/Rendezvous [5], JEDI [6], Siena [7]. There is little application-level QoS support for these system. So they do not satisfy the real-time applications. There are several proposals on the design of publish/subscribe model to improve system performance [8]. DDS provides real-time applications developers with advanced abstract interface as well as QoS parameters [9]. [10] proposed a Real-Time Publish-Subscribe service for Data Distribution Service (DDS) based on Real-Time Event condition. Its critical QoS limitation is that its communication nodes are in a LAN. The real-time QoS in PS paradigm has been focused on application-level requirements [11].
Model for Publish/Subscribe Systems

Figure 1. PS structure.

Figure 2. PS dispatch model.

Figure 1 shows an integration solution that consists of four applications. The publisher (sender) applies a topic-based approach to publish messages to topic A and to topic B. Three subscribers (receivers) subscribe to these topics; one receiver subscribes to topic A, one receiver subscribes to topic B, and one receiver subscribes to topic A and B. The arrows show messages flowing from the publisher to each subscriber according to these subscriptions. In this system, messages are published to "topics" or named logical channels. Subscribers in a topic-based system will receive all messages published to the topics to which they subscribe, and all subscribers to a topic will receive the same messages. The publisher is responsible for defining the classes of messages to which subscribers can subscribe. In such a Publish/Subscribe pattern, sender applications attach each message with the name of a topic, instead of specifying particular receivers. The messaging system then delivers the message to all applications that have asked to receive messages on that topic. Message senders need to build the original message, and leave the task of servicing the receivers to the messaging infrastructure.

Design of A Real-Time Publish/Subscribe System

Packet Real-Time Scheduling Strategy

The design and implementation of a real-time publish/subscribe mechanism is a critical ability to provide data within the required temporal constraints [2]. The objective of system is to find a fair scheduling strategy to satisfy application requirement among different subscribers to the same topic, and appropriate prioritizing among different topics. The design of a packet scheduler depends on many factors such as available resources, instantaneous traffic load, traffic characteristics. To achieve the objectives, the scheduler first decides the time deadline for the packets. Sort the deadline for each packet, transmission should be completed no later than its deadline. If the time is greater than the deadline, then the packet is dropped, otherwise, the packet is sent according to the deadline sequence. By the help of thread priority awareness, Real-time communications can work without affecting publisher or subscriber threads.

Delivery Control

Publish/subscribe messaging is designed by anonymous message exchange between components that produce data (publishers) and those that consume it (subscribers). Each message exchanged has
an associated topic id that pairs together publishers and subscribers. Data issued by a publisher or various publishers for a certain topic is delivered to all subscribers of that topic, allowing variation of components and types of data in the network over time. Publishers and subscribers are decoupled and do not need to know who their peers are; the forwarder is responsible for dynamically completing peer discovery and self-adapting to network changes at execution time. Such mechanism provides delivery timing control. Data issued by a publisher or various publishers for a certain topic is delivered to all subscribers of that topic, providing variation of components and types of data in the network. It remains valid when the data are delivered. And it also provides reliability control. Each subscriber typically requires the ability to specify its own reliability characteristics even when Reliable delivery conflicts with deterministic timing.

**Transmission Parameter**

**Publication Parameters.** Each publication is characterized by four parameters: topic, type, strength and persistence, see Figure 2. The topic is the label that identifies each data flow. The type describes the data format. The strength indicates a publisher’s weight relative to other publishers of the same topic. The persistence indicates how long each publication issue is valid. When the persistence elapses, the subscriber takes the first received publication.

**Subscription Parameters.** Subscriptions are identified by four parameters: topic, type, minimum separation and deadline. The topic the label that identifies the data flow, and type describes the data format (same as the publication properties). Minimum separation defines a period during which no new issues are accepted for that subscription. The deadline specifies how long the subscriber is willing to wait for the next issue.

**The Mathematical Model of Publish/Subscribe Operation**

Three actors participate in PS system: publisher, subscriber, and Information Repository among publish/subscribe (PS) communication. Publishers and subscribers have already elaborated earlier. Information Repository is responsible for defining acknowledgements. There are three types of objects: notification, subscription, and acknowledgements. Let $B$ be a 6-tuple defined as $B = (P, S, I, N, N, A)$ (1), where $P$ is a set of publishers, $S$ is a set of subscribers, $I$ is a set of Information Repositories, $N$ is a set of notifications, $U$ is a set of subscriptions, and $A$ is a set of acknowledgements. $B$ is the structure of publish/subscribe system and defines the boundaries of system’s state space. The three entities presented in the system interact with one another by performing certain action. An event is defined as an action to change operation state. In this case, PS system behaves as discrete event systems. Therefore, let $E = \{e_1, e_2, e_3, \ldots, e_i, \ldots\}$ be a set of events that may occur. Every event $e_i$ occurs at a discrete point in time represented by $t(e_i)$. No two events can occur simultaneously; that is, if $t(e_i) = t(e_j)$, then $e_i = e_j$. Hence, an ordered sequence of events occurs in which $e_i$ always precedes $e_j$ if and only if $t(e_i) < t(e_j)$ given that $i < j$. A publish/subscribe is mathematically defined as $PS = (F, E)$, where $F$ describes the structure of the system and $E$ determines its behavior. PS system behavior can now be modeled as ordered sequence of events which results in change of system states. System state changes when any of the individual publisher, subscriber, and Information Repository changes its state.

**Implementation of Publish/Subscribe Model**

There are low bandwidth and lightweight process requirements for real-time PS. PS was specifically designed to work on top UDP protocol. UDP protocol allows directly sending and receiving messages without establishing any connection. It is better suited for distributed real-time applications, since it is deterministic in the sense of packet retransmission. When a subscriber application sends a message for subscription to the pub/sub server, the pub/sub server keeps the address (IP, port) of the subscriber topic. The Pub/Sub server simply puts a message onto the network with a destination address and hopes that the message arrives. When an event message of a particular topic is sent to a pub/sub server by the publisher application, the pub/sub server takes the address list for the corresponding topic and sends the event to every address of the list. The system
looks up those who subscribe the topic, and forwards the message to them according to the real-time scheduling strategy. A publisher can publish various types of data using Data Writers. Participants create publishers to manage a group of Data Writers. A subscriber is an object responsible for receiving the published data and making it available for the application that the data are supposed to reach. It can receive different types of data using Data Readers. By this way, the application subscribes to the data by name rather than to a specific publisher or publisher location. It can also accommodate configuration changes without disrupting the data flow. By this way, Publish-subscribe offers advantages for real-time applications because it is very efficient in both bandwidth and latency for periodic data exchange, it offers the best transport for distributing data quickly. Because it provides many-to-many connectivity, PS is ideal for applications in which publishers and subscribers are added and removed dynamically.

Conclusion

Some Real-time systems require efficient data delivery. They require rapid process the messaging-oriented task. The use of Publish-subscribe model provides such service with bandwidth utilization. Only minimal delays have been added into the critical data-transfer path. And operations can be event-driven and highly parallelized, thus, a greater transmission gain can be provided than the traditional client-server approach.

Reference