A Reliable UDP for Ubiquitous Communication Environments

Doan Thanh Tran, Eunmi Choi*
School of Business IT
Kookmin University
Jeongnueng-Dong, Seongbuk-Gu, Seoul, 136-702
Korea

Abstract: In the incoming ubiquitous communication environments, a large number of ubiquitous terminal devices tend to communicate with a specific server or other devices that locate locally. This pervasive terminal-initiated communication will become more popularized. In this paper, we propose a reliable communication protocol, called RUDP, that is designed for ubiquitous communication environments where a large number of terminal devices frequently send a short size of requests. The RUPD contains a reliable connection-oriented protocol, which uses a three-way handshaking for connection establishment and a resend mechanism for packet loss, taking advantages of TCP. To observe the performance and reliability, we compare experimental results of RUPD with TCP on top of Java and .NET platforms. The RUDP that we propose in this paper achieves fast and reliable communication compared to an ordinary TCP communication.

Key-Words: **UDP,** Reliable Communication Protocol, Ubiquitous System

1 Introduction

In ubiquitous computing environments [1,9,10] like a digital home, a number of ubiquitous terminal devices tend to communicate with a specific server or other devices that locates locally. As a center of services, the server receives requests from various terminal devices and provides appropriate context-aware services for the requests. To provide context-aware services, the server usually maintains context and persistent information on its local storage. Fast and reliable communication between terminal devices and their server is necessary to provide correct and real-time services [8]. In this paper, we focus on constructing a fast and reliable communication protocol in ubiquitous computing environment.

In addition, in the incoming ubiquitous communication environments, a large number of pervasive terminal devices initiate their communication automatically without human interference. Thus, terminal-initiated communications will become more popularized than human-initiated communications [11]. A large number of terminal devices communicate by embedded short-range wireless communication methods. The packet size of data is relatively small and the terminal-initiated communications become to involve many simultaneous connections to the server. These kinds of requirements in ubiquitous communication environment let us need to have a communication protocol with short and robust connection to a server.

There are many research works on communication protocols by studying the performance of TCP and UDP communications on different networks and OS plat-forms. Jingyi He [3] studied packet aggregation and deflection routing as employed in Optical Packet-switched networks on the performance of upper layer Internet protocols represented by TCP and UDP. George Xylomenos [4] carried out a comprehensive set of measurements of a 2.4 GHz DSSS wireless LAN and analyze its behavior. The issues which were examined are host and interface heterogeneity, bidirectional (TCP) traffic and error Modeling. Andro Milanovic [5] studied data transmission speed on three different operating systems: Windows 95, Windows NT workstation, and Linux 2.1.132. Sherali Zeadally [6] performed some experiments for conventional networking protocols such as TCP- UDP/IP over ATM. Only a few researchers work for reliable UDP areas. E. He [7] studied an aggressive bulk data transfer scheme, called Reliable Blast UDP (RBUDP), intended for extremely high bandwidth, dedicated or Quality-of-Service-

* Corresponding author: Eunmi Choi. This work was supported by the BK21 in 2006. and partially supported by the KOSEF under Grant No. R04-2003-000-10213-0.

enabled networks, such as optically switched networks.

In this paper, we propose a reliable and robust UDP communication mechanism, called RUDP, which is especially for ubiquitous communication environment where a large number of end-terminal devices tend to send request packets with short sizes, such as periodic monitoring data and profile data. The terminal devices initiate the communication to a server and have a connection to the server as the TCP does. However, the terminals do not need to keep their sessions for a long time compared to the ordinary TCP. After showing the protocol of RUDP, we present experimental results of the RUDP on Java and .NET platform environments, compared with the performance results of an ordinary TCP. We also apply various stress conditions in testing scenarios to predict cases of the real-world applications.

This paper is organized as follows. Section 2 describes the target ubiquitous communication environment. Section 3 introduces the RUDP protocol and its characteristics. In section 4, experimental results of performance are presented in several aspects. We conclude in the last section.

2 Target Ubiquitous Communication Environment

As we introduced, ubiquitous communication environment contains a large number of ubiquitous terminal devices as shown in Figure 1. The numerous terminal devices initiate communication to a server automatically without human interference, and send requests to the server, which accepts the requests and processes the requests with the helps of the proper remote service providers on Internet. We consider that the devices tend to move around and the request size is short, such as for sending location information or monitoring a situation in the ubiquitous environment. The terminal-initiated communications are event-driven based communication, so the number of maximum simultaneous connections to the server will be the almost same as the number of terminal devices [11]. As for a different type of stream service re-quests, such as file upload or download and multimedia services, it is necessary to change easily to separate ordinary TCP or UDP communication connections.

In addition, the server needs to accept various requests from different terminal devices, so the communication requires a protocol that can be commonly used. As for context-aware events handling, the packets need to be delivered reliably and the connection to a server needs to be kept in a short-range. Also fast communication is the essential requirement in pervasive ubiquitous terminals. Thus, we need to construct a protocol with fast, reliable, and robust communication. In the next section, we introduce a reliable UDP that is suitable for ubiquitous communication environments.

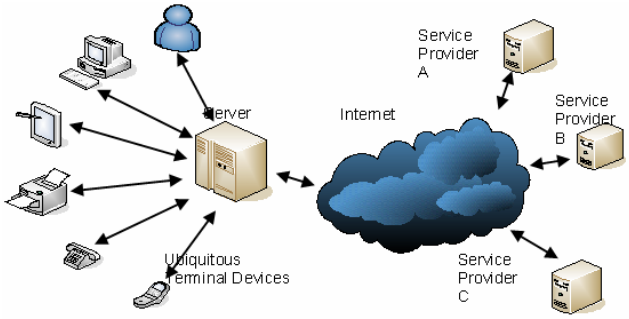

Fig. 1 Ubiquitous Communication Environment

3 Reliable UDP Protocol

In this section, we introduce the Reliable-UDP (RUDP) protocol with its characteristics. We also compare the major characteristics among UDP, RUDP, and TCP.

3.1 The RUDP Protocol

The RUDP protocol we propose has the following characteristics to ensure the reliable delivery between two parties:

- In order to establish a connection, the RUDP uses three-way handshaking with *sessionID* agreement.
- The *sessionID* is used for the further sequences of communication.
- For recovery from a packet loss, the sender resends the same packet after a specific period time if no ACK is received from the receiver.

Figure 2 shows the overall protocol flow between a client and a server. The server performs *a passive OPEN* (*s1*) and is ready to receive *a connection request (CONN)* from a client. The client starts communication by sending a *CONN* message to the server as well as the *SID* of a random number (*c1*), and waits for *ACK* from the server (*c2*). Once the server receives the *CONN control message* (*s3*), it sends an agreement message of *CONN* with another random

number *CHECK* (*s4*). When the client receives the *CONN* control message as the acknowledgement of request (*c3*), the client now sends the data block with acknowledgement for the server's request (*c4*) and the connection is established. The server starts to finalize the connection by sending *FIN* control message (*s7*), and correspondingly the client agrees to finalize the connection by sending the acknowledgement (*c7*). Since this protocol contains three-way handshaking step and the packet resending step if there is no acknowledgement, we adopt the reliable TCP characteristics. However, when the last packet (#5) is lost, the server does not resend the *FIN* message after checking the connection of the client is closed in order to achieve a better performance.

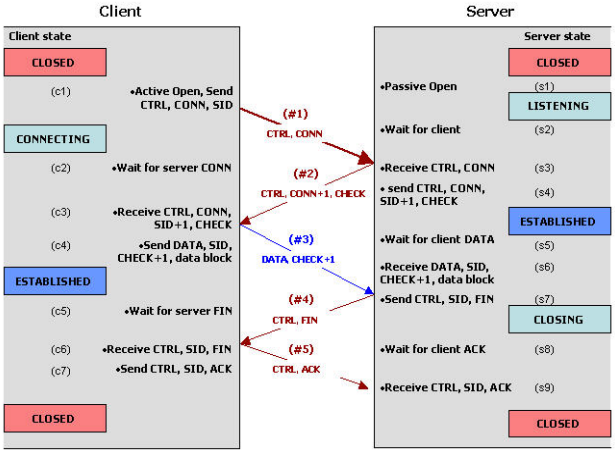

Fig. 2 Protocol Flow of the RUDP

3.2 Characteristic Comparison among the UDP, RUDP, and TCP

In order to compare the major characteristics of UDP, TCP, and RUDP protocols, we list out considerable issues which are valuable in communication Quality of Service criteria [2].

- **Connection-Oriented**: A process of negotiation occurs to establish a connection, ensuring that both communication parties agree on how data is to be exchanged.
- **Bidirectional**: Both communication parties on a connection can send and receive in bidirectional, regardless of which of them initiates the connection.
- **Multiply-Connected and Endpoint-Identified**: It allows each party to have multiple connections opened, either to the same IP or different IP par-ties, and to handle each connection independently without conflicts.
- **Reliable**: This characteristic helps to keep track of data that has been sent and received by ensuring all transmissions send to their destinations.
- **Acknowledged**: It is whether all transmissions are acknowledged, so that it can provide reliability.
- **Stream-Oriented**: This characteristic allows applications to send a continuous stream of data for transmission. Applications don't need to worry about making this into chunks for transmission.
- **Data-Unstructured**: there are no natural divisions between data elements in the application's transmitted data.
- **Data-Flow-Managed**: This ensures that data flows evenly and smoothly, by dealing with problems that arise along the way.

Table 1 shows the comparison of UDP, TCP, and RUDP in terms of the major characteristics of communication explained above. The RUDP supports all characteristics except Bidirectional, Stream-Oriented, and Data-Flow-Managed characteristics. Since these three characteristics are not required in the ubiquitous environment we consider, the RUDP seems to be the best solution of fast and reliable communication to the ubiquitous service environment. As in the table, the RUDP has the property of fast communication as the UDP does, and keeps the reliable property as the TCP does.

Table 1 Comparison of Communication Protocol Characteristics

Characteristics	UDP	R-UDP	TCP
Connection-Oriented	No	Yes and fast	Yes but slow
Bidirectional	No	No	Yes
Multiply-Connected and Endpoint-Identified	No	Yes	Yes
Reliable	No	Yes	Yes
Acknowledge	No	Yes	Yes
Stream-Oriented	No	No	Yes
Data-Unstructured	No	Yes	No
Data-Flow-Managed	No	No	Yes

4 Experimental Results

In this section, we present the experimental results of RUDP communication compared to the ordinary TCP communication. After showing the testing environment, we present the performance of two

protocols varying the numbers of packets, the sizes of packets, and the UDP waiting time. All the performance results are measured on both of Java and .Net platforms.

4.1 Experimental Environment

In order to measure the performance of the proposed protocol RUDP, we set up the following experimental environment. The initial RUDP requesting server is called a client, and the responding receiver is the server. Each client runs in a separate thread and has a separate connection to the server. The connection establishing time and the disconnection time are recorded. All tests run multiple times to assure repeatability and to present the average of them. Performance results are measured under the maximum load by publishing as many messages as possible. Each set of test is performed after rebooting systems. All servers and clients are established before any testing ramp-up periods are begun. All processes are restarted before each test. During the test, no other applications run and use resources of the system. All performance data is collected at beginning and ending of running time.

The system topology consists of two machines: one for executing all clients and the other for executing the server. These systems were interconnected on an isolated network using a single network switch to remove unrelated traffic. Both server and client have the same system configuration as follows:

- Hardware configuration: Intel Pentium 4 3.0Ghz, 1 Gb RAM.
- Platform configuration:
  - Java version: Windows server 2003, Java(TM) 2 Runtime Envi-ronment, Standard Edition (build 1.4.2_04-b04), Java Hot-Spot(TM) Client VM (build 1.4.2_04-b04, mixed mode)
  - .NET version: Windows server 2003, Microsoft .NET Framework version 1.1
- Network setting: Client and server are on the same network segment. 100Mbps Ethernet connection.

4.2 Performance Measurement on Various Numbers of Packets

In order to measure the performance improvement of RUDP compared to TCP, the first set of experiments is designed with various numbers of packets having the same packet size. To generate packets, the client machine creates threads as many as packets generated, and starts to transmit those packets toward the server. Thus, the number of packets is equal to the number of clients connected to the server. By increasing the number of packets, we measure the total elapsed time of transmitting packets. The size of each packet is 1024 bytes and the maximum waiting time is one second, where during the waiting time the RUDP client or server waits for the reply and, if there is no reply, it resends the packet that is previously sent.

Figure 3 shows the experimental results of the total elapsed time as the number of packets increases. As in the figure, the RUDP shows faster from three times up to twelve times than the TCP. Under the heavy traffic, there occurs a few packet losses, but the RUDP resends the packet according the RUDP protocol and processes it correctly without reaching to the omission failure or performance degradation. Also, with the given system environment, the RUDP server usually keeps about 500 connections in a second to process those packets. The results on Java platform are almost similar to those on .NET platform as in Figure 3; results on Java are a few better than those on .Net.

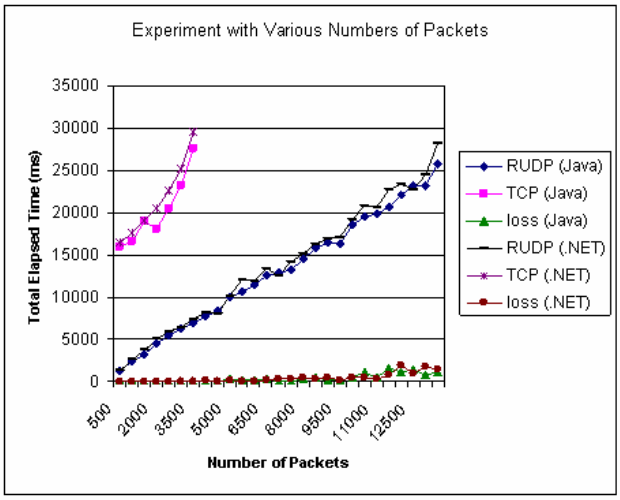

Fig. 3 Performance Results with Various Numbers of Packets

In this experiment, we also find that the TCP has the limitation to establish TCP communication connections with the server. In the TCP communication, a server cannot accept more than 4000 packets, since a server cannot create the large number of sessions. In contrast, the RUDP does not have any limitation to accept packets: the more the clients want to send requests, the more the RUDP can process. This is good to process a huge number of

short requests from a large number of ubiquitous devices in ubiquitous communication environment.

4.3 Performance Measurement on Various Sizes of Packets

The next experiment is to measure transmission time by changing packets sizes. We fix the same number of packets generated. In this scenario, the number of generated packets is fixed to 2000 and the maximum waiting time is 1 second.

Figure 4 shows the performance results by changing the packet sizes. When we use 2000 packets to test, the TCP has all concrete results, compared to the Figure 3. When we send various sizes of packets, the elapsed time of packet transmission does not have any big difference. Over all the range of sizes, the packet transmission time of RUDP is four-time faster than that of the TCP on both of Java and .NET platforms. When the size of a packet is 16384 bytes, the traffic load reaches the limitation by saturating the network bandwidth.

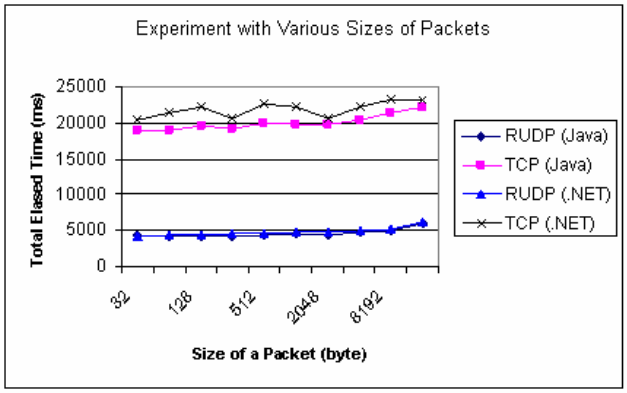

Fig. 4 Performance Results with Various Sizes of Packets

4.4 Performance Measurement on Various RUDP Waiting Time

In this set of experiments, we want to observe the effect of the RUPD waiting time on the overall performance. We change the maximum waiting time of RUDP from one second up to ten seconds in this test set. The size of a packet is fixed to 1024 bytes and we test with two different sets: one with 2000 packets and the other with 5000 packets.

Figure 5 shows the performance results by changing the maximum waiting time of RUDP. Although we change the values of the maximum waiting time from one second up to ten seconds, the total elapsed time does not change a lot in this environment. Thus, choosing a small value of the maximum waiting time is a reasonable choice in the RUDP.

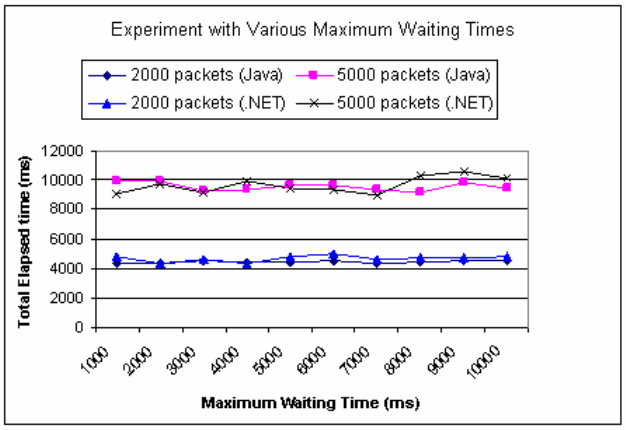

Fig. 5 Performance Results with Various Waiting Times

5 Conclusion

In this paper, we introduced a reliable UDP protocol that is proper for fast and reliable communication in ubiquitous communication environments. The RUDP is a reliable connection-oriented protocol, by taking advantages of TCP, which uses a three-way handshaking for connection establishment and a resending mechanism for packet loss. We performed several sets of experiments to analyze the performance of the RUDP, comparing with that of the TCP. Over all the range of sizes, the packet transmission time of RUDP is more than four-time faster than that of the TCP on both Java and .NET platforms. Compared to the TCP that cannot accept more than 4000 packets, the RUDP does not have any limitation to transmit packets. The RUDP is good to process a huge number of short requests from a large number of ubiquitous devices in the ubiquitous communication environments.

References:

[1] K. Yamazaki, Research directions for ubiquitous services, *Applications and the Internet, 2004. Proceedings - 2004 International Symposium*, 2004, pp.26-30.

[2] TCP/IP Guide, http://www.tcpipguide.com.

[3] J. He and S.H.G. Chan, TCP and UDP Performance for Internet over Optical

Packet-switched Networks, *Communications, 2003. ICC '03 IEEE International Conference*, Vol.2, 2003, pp 1350-1354.

[4] G. Xylomenos and G. C. Polyzos, TCP and UDP Performance over a Wireless LAN, *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol.2, 1999, pp 439-446.

[5] A. Milanovic, S. Srbljic, and V. Sruk, Performance of UDP and TCP Communication on Personal Computers, *Electrotechnical Conference, 2000. MELECON 2000*, Vol.1, 2000, pp 286-289

[6] S. Zeadally, TCP-UDP/IP Performance over ATM on Windows NT, *IEEE ATM Workshop*, 1997. pp 63-72

[7] E. He, J. Leigh, O. Yu, and T. A. DeFanti, Reliable Blast UDP: Predictable High Performance Bulk Data Transfer, *IEEE Cluster Computing*, 2002, pp 317-324.

[8] M.W. Bigrigg, Ubiquitous System Software, *IEEE Pervasive Computing*, Vol.3, No.3, 2004, pp 57-59

[9] U. Saif and D.J. Greaves, Communication primitives for ubiquitous systems or RPC considered harmful, *International Conference on Distributed Computing Systems Workshop*, 2001, pp 240-245

[10] N. Davies and H.W. Gellersen, Beyond prototypes: challenges in deploying ubiquitous Systems, *IEEE Pervasive Computing*, Vol.1, No.1, 2002, pp 26-35

[11] M. Matsumoto and T. Itoh, Study of Server processing Load evaluations in Ubiquitous Communication Environments, *The International Symposium on Applications and the Internet Workshops*, 2004, pp 689-695