A Robust Web-Based Approach for Broadcasting Downward Messages in a Large-Scaled Company

Chih-Chin Liang[1], Chia-Hung Wang[2], Hsing Luh[2], and Ping-Yu Hsu[3]

[1] Telecommunication Laboratories, ChungHwa Telecom Co., Ltd No.21-3, Sec. 1, Sinyi Rd., Jhongjheng District, Taipei City, Taiwan 100, R.O.C.
Department of Management, National Central University, No. 300, Jung-da Rd., Jung-li City, Taoyuan, Taiwan 320, R.O.C.
lgcwow@gmail.com
http://www.chttl.com.tw/eindex.htm
[2] Department of Mathematical Sciences, National Chengchi University, No.64, Sec. 2, Jhihnan Rd., Wen-Shan District, Taipei City, Taiwan 116, R.O.C.
{93751502, slu}@nccu.edu.tw
[3] Department of Management, National Central University, No. 300, Jung-da Rd., Jung-li City, Taoyuan, Taiwan 320, R.O.C.
pyhsu@mgt.ncu.edu.tw

Abstract. Downward communication is a popular push-based scheme to forward messages from headquarters to front-line staff in a large-scaled company. With the maturing intranet and web technology, broadcasting algorithms, including pull-based and push-based broadcasting algorithms, it is feasible to send downward messages through web-based design by sending packets on a network. To avoid losing messages due to the traditional push-based method, companies adopt a pull-based algorithm to build up the broadcasting system. However, although the pull-based method can ensure that a message is received, it has a critical problem, the network is always congested. The push-based method can avoid congesting the network, but it needs a specific robust design to ensure that the message reaches its destination. Hence, adopting only a pull-based or a push-based broadcasting algorithm is no longer feasible especially not for a large-scaled company with complex network architecture. To ensure that every receiver will read downward messages thereby reducing the consumption of network bandwidth, this work proposes a robust web-based push- and pull-based broadcasting system for sending downward messages. This proposed system was successfully applied to a large-scaled company for a one-year period.

Keywords: web-based enterprise systems, e-commerce, downward communication.

1 Introduction

Information management for front-line staff such as front-desk, customer service, call center, and others, is difficult to handle adequately in a large-scaled company. This is due to the fact that the needs of customers change frequently and new services are being generated by a company at a rapid rate in order to meet various demands, e.g. the

K. Aberer et al. (Eds.): WISE 2006, LNCS 4255, pp. 222 – 233, 2006.

promotion for a new product, discounts, etc. A decision maker must disseminate strategic downward messages to the front-line staff through a push-based approach in order to satisfy various demands [1]. Traditionally, the push-based approaches for sending downward messages included official documents, e-mails and telephone messages. However, these methods have defects, such as, messages cannot be ensure to be received, poor efficiency due to the multiple organizational hierarchies of a large-scaled company, and messages passing through too many intermediaries, resulting in misunderstandings [2]. Traditional push-based approaches for sending downward messages are not suitable in a large service-oriented company, because they cannot effectively send downward messages to each receiver.

A broadcasting algorithm used for sending unidirectional packets from senders to numerous receivers through a network is similar to that for sending downward messages. Hence, a company should be able to utilize a broadcasting algorithm to disseminate downward messages, although this has received little attention to-date.. Broadcasting algorithms can be classified into two categories: push-based broadcasting algorithms and pull-based broadcasting algorithms [3]. The push-based broadcasting algorithms are similar to the traditional push-based approaches: a sender actively disseminates a packet to receivers, but has the potential of losing the packet. Repeatedly retrieving a packet from a sender, a pull-based broadcasting algorithm can ensure a packet is retrieved, but it causes a congested network [3].

In recent years, adopting push- and pull-based broadcasting algorithm for sending packets has become a trend for designing a broadcasting system [4], [5], [6], [7]. The present study proposes a robust push- and pull-based broadcasting system (UBS, united broadcasting system) to send downward messages in a large-scaled company with a hierarchical organization. In addition, with the advances made in web-based technology, a front-end staff member can access a server not only through a personal computer but also through various other devices, since UBS is a web-based design [8]. The UBS uses servers as senders and client devices as receivers. Each server allows a client's device to retrieving retrieve messages through a pull-based design, and each message can be replicated among servers through a push-based design. To reduce network traffic for the pull-based design, the UBS client devices retrieve messages mainly from a server in the same local area network (LAN). In addition, to avoid losing messages, a robust push-based design is used to help replicate messages among servers.

To ensure the UBS design is applicable, starting in January 2005 we applied the UBS to the largest telecom company in Taiwan, ChungHwa Telecom (NYSE: CHT), This work simulates the model of UBS to understand the robustness and the efficiency of broadcasting messages to help CHT to decide the scalability of this broadcasting framework. Finally, CHT choose 40 servers as senders to serve more than 10 000 client devices successfully.

The rest of this paper is organized as follows. In Section 2 the broadcasting algorithms and the problems for a robust design to send downward messages are illustrated. In Section 3, the UBS design is described. In Section 4, the derived model of UBS is presented. In Section 5, the managers' utility is discussed. Finally, conclusions are drawn in Section 6.

2 Literature Review

2.1 Broadcasting Algorithm

Broadcasting algorithms can be classified as two methods: the push-based broadcasting algorithm and the pull-based broadcasting algorithm [3]. A push-based broadcasting algorithm is used for the scheduling principle of push-based systems. A sender actively sends items to receivers using scheduling principles. The pull-based broadcasting algorithm indicates that the receiver sends requests to a sender for retrieving items [7].

However, when adopting to send downward messages, it must be noted that each algorithm has its limitation. Adopting the push-based broadcasting algorithm to send downward messages through a network is more effective than traditional approaches and consumes little network resource. However, without a robust design, a push-based broadcasting method potentially has the same drawback as the traditional push-based approaches: the chance of losing messages. In addition, adopting the pull-based broadcasting algorithm can avoid losing messages through repeated request for retrieving messages. However, it places a heavy drain on the network resources [3]. In addition, the network environment of a large-scaled company is a complex internetwork connecting each LAN through a wide-area network (WAN) with restricted network bandwidth. All client devices in the same LAN are connected to each other with unrestricted network bandwidth [4]. Therefore, adopting only a pull-based design or a push-based design cannot fit the requirements of a large-scaled company [5]. Therefore, how to correctly combine the pull-based algorithm and the push-based algorithm becomes key to assist a company sending downward messages.

2.2 The Robust Design of Push-Based Approaches

In the push-based design, all messages must reach a destination. Hence, precise timing and the correct order are the major concerns for broadcasting downward messages, in addition to receiving the correct text of each message [6].

Existing robust push-based broadcasting systems typically utilize three techniques: token-passing approach, discrete acknowledgement approach, and, two-phase approach [10]. The above robust mechanisms can be divided into two kinds of design: adding information to the messages being sent to verify their accuracy and status, e.g. the list of visited stations, and utilizing the signal, and an acknowledgement, to let senders and receivers decide the next action [6].

These robust approaches are focused on ensuring that a destination can receive the completed messages. However, the goals of downward communication are that each active receiver must be able to successfully receive every downward message efficiently, and it must be feasible for the hierarchical network architecture of a large-scale company. Hence, an adjusted robust design is needed.

3 The Design of the UBS

The UBS is designed as a hierarchical architecture, where a sender connects its parent sender and its child senders through WAN. A receiver mainly connects a sender within

the LAN. Fig. 1 shows the architecture of the UBS. The chief sender (CS), the senders, and the receivers have a message storage in which each broadcast messages can be stored.

The design of the UBS makes the following assumptions: at least one sender is in operational mode, the clocks of all senders are synchronized, and a downward message is equally distributed throughout a balance tree in which the depth and the number of layers in each branch are the same. The operational mode indicates that a sender, the CS, or a client is functioning well, and that the message storage can be accessed successfully.

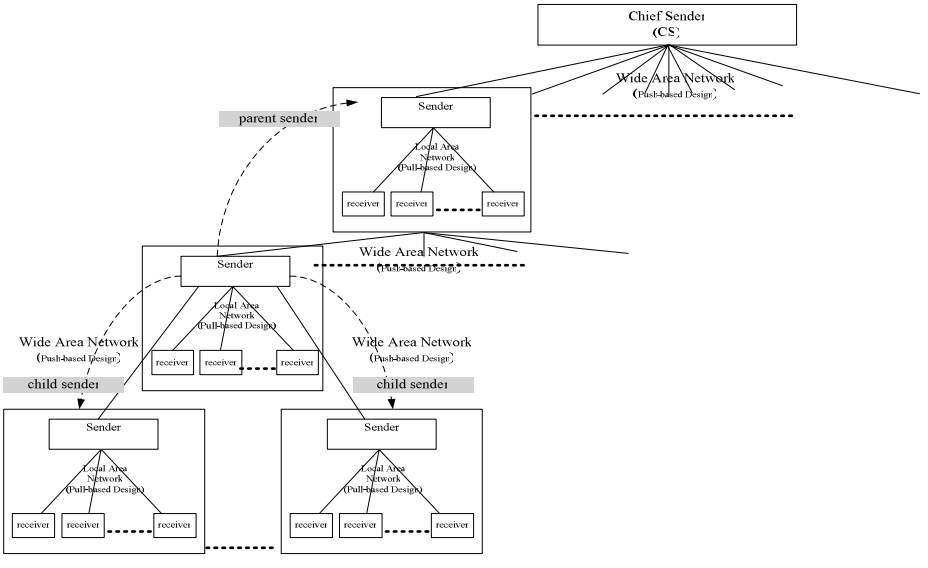

Fig. 1. The architecture of the UBS

To avoid the frequent transfer of messages between each sender through WAN, transmitting messages among senders using a robust push-based broadcasting design. Utilizing a pull-based broadcasting design to transfer messages between a sender and a receiver in the same LAN can reduce the consumption of network resources. In addition, the UBS utilizes the designs of robust push-based broadcasting mechanisms to ensure a message can be replicated among servers: hiding information in a downward message, and sending a specified signal or acknowledgement regarding the status of a downward message.

The data structure of a downward message with hidden information consists of the text, the timestamp, and the message length. The timestamp records the time point in a transaction maintained by the computer in fractions of a second, and is used for various synchronization goals. The message length is the number of characters of the text for verifying the accuracy of a received message.

Additionally, to ensure each child sender receives unread messages, a sender sends a signal to inform all its child senders about a new message that it is ready to send before

it sends a message to its child senders. Hence, if a child sender receives a signal about a new message being sent, but receives no message within a certain time interval, then the sender sends a signal requesting that another sender sends the unread message to the sender. Moreover, if a sender is unable to send a new message to a child sender, the sender then sends a signal about the message it cannot send to the sender to the rest of the child senders. Therefore, the rest of the child senders can send the message to the child senders of the sender without the message.

3.1 General Procedure

An operator writes a message and sends it to a sender A. Timestamp and message length are generated whenever the text of the message is written to the message storage of sender A. Next, sender A sends a signal asking to send the downward message to the CS and to all child senders of sender A. After the request is received, the CS and each child sender of sender A are locked in sequence. Then, sender A sends the downward message to the CS and to each of its child senders. Next, the downward message is recorded in the message storages of the CS and every child sender of sender A, based on the timestamp sequence. Next, the CS and all child senders of sender A broadcast the downward message until the CS and all senders in operational mode receive the complete message. The process of broadcasting a downward message is similar to the above steps when a broadcast operator writes a message and sends it directly to the CS.

Each receiver frequently sends a request for a new downward message to the sender in the same LAN, and retrieves the unread downward message from the sender within the same LAN. To compare the most recent timestamp of the stored downward message of a receiver and the timestamp of the new downward message, the new downward message is stored in the message storage of a receiver in timestamp sequence and displays the message text to a front-line staff member.

3.2 Recovery Procedure

Potential process and communication failures are as follows: crash [12], general omission [12], and timing failure [13], [14]. However, since the robust design of UBS assumes synchronized time, timing failure is not discussed in this paper. In addition, all handling phases of computer errors are classified as follows: error detection; damage assessment; error recovery; and, fault treatment and continuing service [15].

In the error detection phase, a sender and a receiver have two possible error situations: crash and general omission. In the damage assessment phase, a crash indicates that a sender or a receiver is not working properly and transmits no message. Additionally, if a sender general omission is detected, all child senders of the sender will incorrectly receive a downward message. A receiver retrieves the incomplete downward message from a sender when a general omission occurs.

In the error recovery phase, to run an error recovery, a sender or a receiver adopts the forward recovery for a crash and general omission. The sender reserves its current status and cancels the request for retrieval messages from receivers until the error situation is eliminated. When the sender recovers from a crash, it sends a signal of request consisting of a timestamp and an IP address to the parent sender. Unread downward messages are sent to the recovered sender after the parent sender compares

the received timestamp with the latest timestamp for the stored downward message in its message storage. When a general omission occurs, storing an uncompleted downward message in the message storage is deemed incorrect. Furthermore, no receiver in the same LAN as the sender in which the general omission occurred can retrieve downward messages from the sender. Finally, to ensure that unread downward messages can be received, the recovered sender sends a request to its parent sender to ensure that the parent sender resends the unread downward messages. Additionally, in the error recovery phase, the receiver retains its status and retrieves new downward messages when it recovers from a crash. To recover from a general omission, the receiver retrieves the downward message again since the received message is incomplete.

In the fault-treatment and continuing service phase, when a sender crashes, the downward message cannot be stored in the message storage within a certain time interval. A sender resends the lost downward message whenever the crashed parent sender recovers. However, the broadcast service must continue even when a crash occurs. A parent sender sends a signal to ask that a new downward message be inserted into its child senders, before sending the new message. For example, sending no downward message to sender A, the parent sender of sender A instead sends the downward message to the other child senders. Moreover, the notice, "sender A has malfunctioned" is also sent to the rest of the child senders. Hence, the rest of the child senders are in charge of broadcasting the downward message to all child senders of sender A. In addition, if a sender cannot send the new inserted message to the CS, the sender sends the message to the senders on the second layer and continues the broadcasting service. Whenever the CS recovers, it compares the stored messages with the senders on the second layer and requests the resending of the unread messages to the CS.

However, a sender may crash before the downward message is completely sent to its child senders. That is, not all child senders may receive the downward message. A robust solution to this problem is as follows: whenever the downward message is stored in the message storage of a sender, the signal, "the new downward message is broadcasted," is broadcasted to all child senders. Within a certain time interval, each child sender of the sender in which the downward message is not received then sends a request to the sender to resend the downward message. If a child sender of the crash sender still does not receive the downward message within the certain time interval, it sends a request to another sender, until the downward message has been successfully received. At least one sender is then in operational mode, and the child senders of the crashed sender can find at least one sender to send the downward message. Hence, the procedure of broadcasting downward messages continues to work properly.

Moreover, when a general omission occurs, the downward message cannot be stored in the message storage of a sender. The sender then sends a request for the downward message to be resent to its parent sender. In addition, if a sender then still does not receive a downward message from its parent sender, then the sender sends the request to other senders to retrieve the downward message.

The robust design of the UBS can also deal with mixed errors, crashes and general omissions. If the parent sender crashes, it will produce a general omission. If the parent sender is first omitted and then crashes, then the incomplete downward message is not stored in the message storage of the parent sender. Hence, the child senders will not receive an incomplete downward message. If a child sender crashes, then it cannot generate a general omission. If a child sender is first omitted and then crashes, the

incomplete downward message is not stored in its message storage. Hence, no incomplete downward message is delivered to any sender. However, if a sender receives a new downward message that is incorrect and the parent sender crashes before the request to resend the downward message is received, then this robust design can accommodate the scenario. For example, a sender sends a request to send a downward message to another sender. It waits for a certain time interval and receives no downward message. If a sender has already sent the request to send downward message to its parent sender, then the parent sender crashes immediately before a new downward message is sent, the sender then sends a request to send a downward message to another sender after waiting for a specific time interval and receiving no downward message.

In the fault-tolerant and continuing service phase, a receiver retrieves no new message until it recovers from the crash. Whenever a receiver is in operational mode and cannot find a parent sender to connect to, it will try to connect to one of the grand senders instead.

In summary, this design is robust and efficient since a receiver can correctly receive messages and the push- and pull-based broadcasting technology in this design can work with acceptable bandwidth consumption.

4 A Push- and Pull-Based Model

In this section, the steady state probability will be studied to learn the mean waiting time and the probability of no server in operational mode. Additionally, adopting extra servers, the senders, can improve the performance of sending downward messages but the cost is arising. Hence, deciding the proper number of servers under budget constraint is a major problem of the broadcasting system. The time interval of sending each message is a random variable. That is, the arrival time is a random variable. In this system, clients, the receivers, ask to retrieve messages from a server repeatedly and the period of a message retrieved by every client is assumed continuity.

4.1 Problem Formulation

Clients can retrieve new downward messages one by one. The broadcasting process is finished after the last client gets the message. For example, 10,000 clients try to get a downward message means 10,000 jobs arrive this queueing system and wait for service. Assuming the arrival stream has k jobs, k client computers, the actual number of jobs in an arriving module is k.

There are s machines (servers) in the system to send downward messages, and they have i.i.d. exponential lifetimes. Let $1/\alpha$ be the mean time to the failure, include crash and general omission, of a machine. When a machine fails, it needs mean repair time $1/\beta$.

We assume that both inter arrival and service times follow exponential distribution with means λ^{-1} and μ^{-1} respectively. A state of system is denoted by (n,i), where n is the number of jobs in the service or in the waiting room, $n \geqq 0$, and i is the number of servers in operational mode in the system, $0 \leqq i \leqq s$. The stationary probability is denoted by $\boldsymbol{\pi}$ as

$$\boldsymbol{\pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots),$$

where $\boldsymbol{\pi}_{\mathbf{n}}$ is a $(s+1)$-dimensional row vector of stationary probability when there are n jobs in the system. For this system, a steady state will exist if the utilization factor $\rho < 1$. It is clear that the queueing problem is still Markovian in the sense that future behavior is a function only for present not for the past. Let $W(s)$ be the mean waiting time in the system given s servers initially. We want to find the steady-state probability, P_n, when n jobs in the system.This work arranges the states (n, i) in lexicographic order and partition of the state space according to the number of customers, n, i.e.

$$S_n = \{(n,i) \mid 0 \le i \le s\},\ n = 0,1,2,\ldots$$

In this model, we have the transition rate matrix $\boldsymbol{Q}$ which is of the block form. For the state balance equations $\boldsymbol{\pi Q} = \mathbf{0}$ and the normalization condition $\boldsymbol{\pi 1} = \mathbf{1}$, we can compute $\boldsymbol{\pi}_n$, $n = 0,1,\ldots$. Then we have the steady state probability

$$P_n = \boldsymbol{\pi}_n \cdot \mathbf{1} \tag{1}$$

and the limiting probability that there are i servers in operational mode

$$P((i-1) \text{ servers in operational mode}) = \left(\sum_{n=0}^{\infty} \boldsymbol{\pi}_n \right) \cdot e_i, \tag{2}$$

for $i=1,\ldots,s+1$, where e_i is a $(s+1)$-dimensional column vector with one at the i-th component and zero elsewhere. In specific, we have the limiting probability that there is no server in operational mode in the system,

$$P(\text{no server in operational mode}) = \left(\sum_{n=0}^{\infty} \boldsymbol{\pi}_n \right) \cdot e_1, \tag{3}$$

and this probability is small enough to ignore for $\alpha \ll \beta$. From equation (1), we can compute the mean number of jobs in the system,

$$L = \sum_{n=0}^{\infty} n P_n. \tag{4}$$

By using the Little's formula, we also have mean waiting time

$$W(s) = \frac{L}{\lambda}. \tag{5}$$

With mean waiting time W, our objective is to determine the increase in investment of service for a corresponding decrease in system delay.

4.2 Queueing Design Problems

This work considers a system with unknown number of servers s, where s belongs to the set of positive integers Z^+. Assuming a set-up cost M is for each server. Suppose the company has the limited budget B for adding servers. Our objective is to determine the optimal value of s under the budget constraint. If there are s servers in the system, then

k jobs are shared by servers. This work assumes k jobs are divided equally by s servers for simplicity, From equation (5), we have the average waiting time in the system $W(s)$.

Using $U(W(s))$ to denote the manager's utility function for expected waiting time $W(s)$. The expected waiting time $W(s)$ will decrease when the number of servers s increases. Assume the manager's utility $U(W(s))$ increases as decreasing $W(s)$. The objective is to maximize the objective function $f(s)$ with respect to s, where

$$f(s) = U(W(s)) - M \cdot s. \tag{6}$$

We assume $f(s)$ is unimodal of s on $[0,B/M]$.This work uses this representation to describe the manager's choice. The manager will choose so as to maximize his or her utility function on the feasible set. The problem of maximizing a given function on a given set is a mathematical problem. This gives rise to the following optimization problem

$$\begin{aligned} \max\ & f(s) = U(W(s)) - M \cdot s \\ \text{st.}\ & M \cdot s \le B \\ & W(s) \le D \\ & s \in Z^{+} \end{aligned} \tag{7}$$

where D is the delay constraint. If s^* is the optimal number of servers under the budget constraint, then the basic geometric properties of this objective function $f(s)$ are that it is increasing as $0<s<s^*$ and it becomes decreasing as $s \geqq s^*+1$.

The efficiency of the computations relies not only on the implicit enumeration scheme but also on the efficiency of calculations of the probabilities. Due to the complexity of the probabilities and the requirement of integer values for the decision variables, a search technique such as implicit enumeration is usually required. A set of decision variables must first be specified, and then the constraints and objective function are evaluated. This optimization problem gives rise to a trade-off of better customer service versus the expense of providing more service capability.

5 Results

5.1 Implementation

The best way to prove that this design is applicable is by successfully applying it to a large-scaled company. The largest service-oriented company in Taiwan, ChungHwa Telecom Company, CHT, has more than 10,000 front-line staff members spread out over Taiwan. As of 2003, CHT had built up a pull-based broadcasting system where one server connects to 10,000 client devices. However, this had two drawbacks: a congested network and the ability to send only one downward message at once.

Finally, CHT decided to adopt a self-developed web-based broadcasting system to replace the original design. The present study proposes a novel design with push- and pull-based broadcasting algorithms. To study the acceptable number of servers for CHT, this work simulates the model proposed in section 4 for a period of one year in order to determine the average waiting time for broadcasting messages. The following

set of parameters is derived from the experience of the original design of CHT. The mean time of sending one message from a server to a destination is 0.0082 second, 217.4 downward messages need to be broadcast each day, the mean time between machine failure is $1/\alpha$= 1,094.4 hours, and the average repair time for each malfunction is $1/\beta$= 6.4 hours. That is, the probability of no server being in operational mode, equation (3), is small enough that it can be ignored, since $\alpha << \beta$. Moreover, the working time of the system is 24 hours a day, seven working days a week, and 52 weeks a year. To fit the organizational hierarchy, this work assumes that one server is set up at each center and at headquarters. The organization hierarchy of CHT has five layers, the first layer is the headquarters, the second layer has three regional administrative centers, the third layer has 36 regional centers, the fourth layer has 144 service centers, and the fifth layer has 10,000 front-end staff members. The present study simulates the model from two layers (the first layer and the fifth layer, making for one server for 10,000 clients) to five layers (including the servers from all five layers, resulting in 184 servers for10,000 clients) for a period of one year.

CHT's budget for this project was $1,100,000 US dollars. Experience tells us that the cost for installing an extra server is $5,882 US dollars per server and the cost of setting up the network environment for operating the UBS is $8,825 US dollars. This work proposes a simple survey to learn the users' utility. It presents four sets of simulated results and investment cost, from one server to 184 servers, to 184 executive managers for all of the units of CHT. In the present study, the managers are given 10 points each and they are asked to assign these 10 points among the users' utility. Table 1 shows the simulated average waiting time, real investment cost, and evaluated users' utility score.

Table 1. The table of average waiting time, investment cost and users' utility

Number of servers	The average waiting time to broadcast a message (seconds)	Investing cost (US dollars)	Users' Utility
2-tier architecture with 1 server and 10 000 clients	85.23	$ 14,707	1.31
3-tier architecture with 4 servers and 10,000 clients	35.34	$ 32,353	2.69
4-tier architecture with 40 servers and 10 000 clients	16.79	$ 244,105	4.44
5-tier architecture with 184 servers and 10 000 clients	11.90	$ 1,091,113	1.56

As the described by the above results, CHT decided to set up a four-tier broadcasting system with 40 servers and 10,000 client devices allocated based on the organizational hierarchy. Finally, the UBS successfully helped CHT send downward messages to front-line staff members for the "2005 Spring Multi-media Computer Exposition." The salespersons of CHT used different types of equipment to retrieve messages through the web-based application [17].

6 Conclusion and Remarks

Downward communication is an important way to send strategic messages to front-line staff members. However, to do this in an efficient manner is rather difficult to do, especially for a large-scaled company.

Although network technology can be utilized for broadcasting downward messages, simply using pull-based or push-based broadcasting algorithms is not adequate for a complex internetwork. A large-scaled company adopts a pull-based broadcasting algorithm to send downward messages to numerous client devices through one server only. However, the accompanying problems are a congested network traffic and an inefficient broadcasting mechanism.

This work proposed UBS, a robust web-based push- and pull-based broadcasting system. The largest telecom company in Taiwan, CHT, decided to utilize UBS as the broadcasting system and they adopted a four-layer architecture consisting of 40 servers serving over 10,000 client devices into 40 groups since January 2005.

However, it should be noted that our design has one issue that needs to be solved and requires further discussion. That is, if a new message is successfully sent to a sender, and if that sender then crashes immediately before the signal of "new message received" can be broadcast to its child senders, then that causes the problem that the new message can not be broadcast to the child senders. Because child senders are unable to be notified that a new message is coming, they also can not send any request to other senders to send the message again. Fortunately, the above situation has never happened since the system was implemented. The possible reason for this may be that the time interval is extremely short between the message being received completely to the signal being sent out to every child sender.

Acknowledgement. This study is supported by National Science Council, Taiwan, Republic of China, through the Project No. NSC 95-2416-H-008-028.

References

1. Agrawal, M., Rao, H. R., Sanders, G. L.: Impact of Mobile Computing Terminals in Police Work**.** J Organ Comput El Commer, 13(2003) 73-89.
2. Greenberg , J., Baron, R. A.: Behavior in Organizations, Allyn and Bacon Inc, MA (1993).
3. Défago, X., Schiper, A., and Urbán, P.: Total Order Broadcast and Multicast Algorithms; Taxonomy and Survey. ACM Comput Surv, 36 (2004) 372-421.
4. Herrería-Alonso, S., Suárez-González, A., Fernández-Veiga, M., Rubio, R. F. R., López-García, C.: Improving aggregate flow control in differentiated services networks. Comput Net, 44 (2004) 499-512.
5. Wang, W. M., Liang, C. C., Lu, H. Z., Chow, W. S., Chang, K. Y.: Research of Testing Process: The Case of TOPS-System Delivery Process. TL Tech J, 34(2004) 7-34.
6. Saxena, N., Pinotti, C. M., Das, S. K.: A Probabilistic Push-Pull Hybrid Scheduling Algorithm for Asymmetric Wireless Environment. IEEE conference GLOBALCOM (2004) 5-9.
7. Bhide, M., Deolasee, P., Katkar, A., Panchbudhe, A., Ramamritham, K., and Shenoy, P.: Adaptive Push-Pull: Disseminating Dynamic Web Data. IEEE Trans Comput 51 (2002) 652-668.

8. Blake, M. B., Fado, D. H., Mack, G. A.: A publicsh and subscribe collaborateon architecture for web based information. ACM conference WWW (2005) 1164-1165.
9. Rodeh, O.,Birman, K. P., Dolev, D.: Using AVL trees for fault-tolerant group key management. J Inform Sec, 1 (2002) 84-99.
10. Birman, K. P. and Renesse, R.: Reliable Distributed Computing with the Isis Toolkit. Wiley: IEEE Computer Society Press (1994).
11. Lu, H. Z., Liang, C. C., Chuan, C. C., Wang, W. M.: Discussion of TOPS/Order Software Dissemination, news publish and operation mechanism. TL Tech J, 42 (2005) 719-733.
12. Fich, F., Rparentt, E.: Hundreds of impossibility results for distributed computing. Dist Comput, 16 (2003) 121-163.
13. Davies, D. W.: An Historical Study of the Beginnings of Packet Switching. Comput J, 44 (2001) 152.
14. Androutsellis-Theotokis, S., Spinellis, D., A survey of peer-to-peer content distribution technologies. ACM Comput Surv, 36 (2004) 335-371.
15. Castro, M., Liskov, B.: Practical Byzantine Fault Tolerance and Proactive Recovery. ACM Trans Comput Syst, 20 (2002) 398-461.
16. Kamilla, K., Håkan, L., Lundberg, L., Svahnberg, C.,: Optimal recovery schemes in fault tolerant distributed computing, Acta Inf, 41(2005) 341-365.
17. Pascal, C. L.: Enabling Chance Interaction Through Instant Messaging**.** IEEE Trans Prof Commun, 46 (2003) 138-141.
18. Liang, C. C., Hsu, P. Y., Leu, J. D., Luh, H.,: Industrial Track: An effective approach for content delivery in an evolving intranet environment- a case study of the largest telecom company in Taiwan, Lect Notes Comput Sci, 3806 (2005) 740- 749.