Future Generation Computer Systems 26 (2010) 318–323

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Real-time performance analysis for publish/subscribe systems

Sangyoon Oh [a,*], Jai-Hoon Kim [b,1], Geoffrey Fox [c,2]

[a] Division of Information and Computer Engineering, Ajou University, Suwon, South Korea
[b] Graduate School of Information and Communications, Ajou University, Suwon, South Korea
[c] Pervasive Computing Labs., Indiana University, Bloomington, IN, USA

ARTICLE INFO

Article history:
Received 2 October 2008
Received in revised form
18 February 2009
Accepted 21 September 2009
Available online 24 September 2009

Keywords:
Real time
Performance analysis
Publish–subscribe

ABSTRACT

The publish/subscribe communication system has been a popular communication model in many areas. Especially, it is well suited for a distributed real-time system in many ways. However, the research of cost model and analysis of publish/subscribe system in a distributed real-time system have not been suggested yet. In this paper, we present our cost model for publish/subscribe system in a real-time domain, analyze its performance, and compare it with other communication models such as request/reply and polling models. Our empirical result on mobile embedded device shows accordance with cost analysis, which verifies correctness and usefulness of our cost model.

1. Introduction

The publish/subscribe communication system [1] has been popular in many distributed application domains. Unlike traditional point-to-point model such as client–server, publish/subscribe model decouples publisher and subscriber in time, space, and synchronization. Producer (i.e. event source) declares the topics on which they intend to publish event (data) and subscriber (i.e. event displayer) register the topics of interest. When the producer publishes events on a topic, server (i.e. event brokering system) disseminates events to the subscriber. Subscriber can access published data asynchronously anytime and anywhere at its own convenience. Because of its location transparency and flexibility to dynamically add and remove participants, it is an appropriate communication system for large scale loosely coupled distributed systems. Examples of such systems include collaboration systems which require an asynchronous multicast messaging system and military system requiring a distributed real-time system support.

The Publish/subscribe systems are well suited for distributed real-time systems in a number of ways [2,3]. First, events are delivered to the subscribers immediately after the event occurrence, thus subscriber can access the event data in real time. Second, it is asynchronous. The Publish/subscribe systems free the data sender (publisher) from waiting for an acknowledgement by the receiver (subscriber). Thus, publisher can quickly move on to the next receiver within deterministic time without any synchronous operations. The other benefit of having publish/subscribe system for a distributed real-time system is its multicast-like model. Publisher sends only one event to the event broker and the event is delivered to many subscribers [4]. Thus, an increasing number of distributed real-time systems adopt publish/subscribe system for data transfer among a massive number of distributed entities.

There are many challenges related to Peer-to-Peer (P2P) systems. The performance and availability of the P2P system are greatly affected by its underlying interaction scheme. Both Structured P2P (e.g. Tapestry [5], NaradaBrokering [6], and SCRIBE [7]) and Unstructured P2P (e.g. Gnutella [8] and Kazaa [9]) require updated peer information while it is running. Whether it is using centralized architecture with the Distributed Hash Table or it is using the Gossiping method, during their organization of information process (e.g. routing, storage, and discovery) requires the deadline to deliver and to provide the up-to-date information to the overall system. We believe that publish/subscribe based P2P system can perform better than the traditional interaction scheme such as the client–server interaction. Our real-time performance analysis and simulation result are well suited to the proposition and can be used to support it.

There are many advantages to use a publish/subscribe communication paradigm for real-time applications. As noted above, its loosely coupled nature and multicast-like notification is best fitted to the domain. To verify the advantage formally, it is required to have an analytical model and validate it through the simulation and/or the empirical experiment. Even though there has

* Corresponding author. Tel.: +82 31 219 2633.
E-mail addresses: syoh@ajou.ac.kr (S. Oh), jaikim@ajou.ac.kr (J.-H. Kim), gcf@indiana.edu (G. Fox).
[1] Tel.: +82 31 219 2546.
[2] Tel.: +1 812 219 4643.

0167-739X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2009.09.001

Table 1
Model selection.

	Models		Remarks
	Publish/subscribe	Request/reply	
Number of node	Large	Small	Pub/submodel has advantage when system is large and data transfer is shared among many clients.
Number of event (data update) per client's access	Small	Large	Pub/submodel is appropriate when events or data update occurs infrequently.
Access rate of client	High	Low	When clients seldom use published data, pub/submodel is not appropriate.
Degree of common interest	High	–	Pub/submodel is appropriate to disseminate data of common interest.
Cost of user's intervention (pull based)	High	–	Pub/submodel requires less user's intervention than request/reply model.
Delay cost of event (data) transfer to user	High	–	Events (data update) are immediately delivered to subscribers.
Real-time performance	Hard deadline Short deadline	–	Pub/submodel has advantages especially when deadline is short or strict.

been a lot of research proposals and implementations of publish/subscribe communication model [10–14] to improve performance of the system including Siena [15], Gryphon [16], JEDI [17], Rebeca [18], Elvin [5] and [22]. However, to the best of our knowledge, research on performance modeling for a distributed real-time system using a publish/subscribe system has not been announced yet. We propose cost model for general publish/subscribe systems and publish/subscribe system in distributed real-time systems, analyze their performance, and compare them to other interaction based models such as client–server model and polling models. We can estimate the performance of a publish/subscribe system in a distributed real-time system. We can also effectively adopt publish/subscribe systems by using our proposed cost model and analysis of publish/subscribe systems.

Networking middleware that implements a real-time publish/subscribe model such as Data Distribution Service (DDS) [19] is being used for many application domains. It targets high performance (e.g. low latency, high throughput) applications, such as multimedia or military systems. Thus, it is getting important to analyze performance and effectiveness of a publish/subscribe communication system, especially in condition in which time is a key parameter such as in real-time condition.

Our analysis shows that we can choose model as follows by a rule of thumb. As shown in Table 1 and results of our analysis, publish/subscribe model are effective in many cases. For the analysis, we define pull based publish/subscribe system model as the case when the user has an intention to retrieve data (or message) from the broker.

We also experimentally measured and compared the performance of publish/subscriber model to client/server model on our test bed with NaradaBrokering which is a publish/subscribe based message brokering system to verify correctness of our performance model on the real systems. Our cost analysis model is simple but accordant with experimental results.

2. Cost model

2.1. System models

In this subsection, we propose cost analysis model for publish/subscribe systems. We assume the following basic system parameters to analyze cost.

- α (publish rate): We assume that publisher's event generation is governed by Poisson process with average inter arrival time of $1/\alpha$.
- β (request rate or process (reference access) rate): We use this parameter for two meanings: (1) subscriber's access rate of published events, and (2) request rate of client in the client/server models. We assume that these rates are also governed by Poisson process.
- c_{ps} (α) (publish/subscribe cost per event): Cost required for an event publish. c_{ps} is divided into two parts: (1) c_{pub}: ES (Even Source) publish events to EBS (Event Brokering System), and (2) c_{sub}: EBS (Event Brokering System) relays the events to ED (Event Displayer) which registered for the events.
- c_{rr}: (β) (cost per request and reply): Cost for sending request and receiving response in client–server model.
- $c_{poll}(\alpha, T)$ (cost of periodic publish or polling): We assume function of α and T (ex. $cp\alpha T$, c_{poll}), where T is the length of period. (We can also think it as a cost of periodic polling in client/server model.)
- $c_d(\alpha, T)$ (cost of delaying publish): It is cost (or penalty) by delaying data transfer. We assume function of α and T (ex. αT). We need to assign some function for each application.
- $s(n)$ (effect of sharing among n subscribers): For example, server can deliver events with low cost when it broadcasts event to many subscribers. It will be between $1/n$ and 1.
- t_{ps} (time delay for publish/subscribe): Time delay for publishing an event. t_{ps} is divided into two parts: (1) t_{pub}: time delay for publish, ES (Even Source) publish events to EBS (Event Brokering System), and (2) t_{sub}: time delay for subscribe, ED (Event Displayer) subscribes events from EBS (Event Brokering System).
- t_{rr} (time delay for request and reply): Time delay required for sending request message and receiving response message in request–reply (client–server) model.
- $t_{poll}(\alpha, T)$: Time delay for periodic publish.
- D: Relative deadline from user's access intention or event occurrence.

2.2. Cost analysis

In this analysis, we analyze cost of three different models, publish/subscribe, request/reply, and periodic polling models without any failure of communication link or node. We consider (1) conceptual total cost (e.g., the number of message, amount of message, or time delay) per unit time for each model, (2) cost for each access by client (or subscriber), (3) time delay for access after subscriber's (or client's) intention, and (4) time delay between event occurrence and notification to subscriber (or recognition by client). Cost can be the number of message, amount of message, or time delay.

2.2.1. Cost of publish/subscribe model

Since we assume that c_{pub} is cost for that ES (Even Source) publish events to EBS (Event Brokering System), and c_{sub} is cost for that ED (Event Displayer) subscribes events from EBS (Event Brokering System), cost of publish/subscribe model for each event publish and subscribe is $c_{pub} + n\ s(n)c_{sub}$. Please remember that n is the average number of subscriber and $s(n)$ is sharing effect among n nodes. When publish rate is α, cost per time unit is:

$$\alpha(c_{pub} + n\ s(n)c_{sub}).$$

Table 2
Cost analysis for different models.

Model	Publish/subscribe	Request/reply	Polling
Conceptual total cost per time unit	$\alpha(c_{\text{pub}} + n\, s(n) c_{\text{sub}})$	$\beta n\, c_{\text{rr}}$.	$(c_{\text{poll}}(\alpha, T) + c_{\text{delay}}(\alpha, T))/T$
Cost for each access	$\frac{\alpha}{\beta}\left(\frac{c_{\text{pub}}}{n} + c_{\text{sub}}\right)$	c_{rr}	$c_{\text{poll}}(\alpha, T) + c_{\text{delay}}(\alpha, T)$
Time delay between intention and access	0	t_{rr}	$T/2$
Time delay between event occurrence and notification/recognition (or access)	$t_{\text{ps}} = t_{\text{pub}} + t_{\text{sub}}\left(t_{\text{ps}} = t_{\text{pub}} + t_{\text{sub}+}\frac{1}{\beta}\right)$	$\frac{1}{2\beta}$	$T/2$
Deadline meet ratio from user's access intention	1	1 when $D \geq t_{\text{rr}}$ 0 when $D < t_{\text{rr}}$	1 when $D \geq T$ D/T when $D < T$
Deadline meet ratio from event occurrence	1 when $\geq t_{\text{ps}}$ 0 when $D < t_{\text{ps}}$	$1 - \varepsilon^{-\beta D}$	1 when $D \geq T$ D/T when $D < T$

Now, we consider cost in the view point of subscriber (per each event access of subscriber). We analyze three performance metrics, (1) conceptual cost for each access, (2) time delay for subscriber to access event after its intention, (3) and time delay until notification to subscriber after event occurring. The average number of event occurred before each access is cost for each access:

$$\sum_{i=0}^{\infty} \frac{\beta}{\alpha+\beta}\left(\frac{\alpha}{\alpha+\beta}\right)^{i} = \frac{\alpha}{\beta},$$

where c_{pub} is shared among n subscriber and c_{sub} is required for each subscriber. Thus, average cost for each access is:

$$\frac{\alpha}{\beta}\left(\frac{c_{\text{pub}}}{n} + c_{\text{sub}}\right).$$

There is no time delay for access after subscriber's intention since event has already been received. Time delay between event occurrence and notification to subscriber is:

$$t_{\text{ps}} = t_{\text{pub}} + t_{\text{sub}}.$$

We analyze real-time performance (deadline meet ratio) for two aspects: one relative deadline (D) is set from the subscriber's intention to access data and the other deadline is set from the occurrence of event. Deadline meet ratio from the subscriber's intention is always 100% since data was published to the subscriber before subscriber intends to access. However, deadline meet ratio from the occurrence of event is different. When $D \geq t_{\text{ps}}$, subscriber can access data (event) within the deadline. However, when $D < t_{\text{ps}}$, subscriber cannot access data (event) within the deadline.

Deadline meet ratio from the subscriber's intention is 1.

Deadline meet ratio from the occurrence of event is:

$$\begin{aligned} &1 \quad \text{when } D \geq t_{\text{ps}} \\ &0 \quad \text{when } D < t_{\text{ps}}. \end{aligned}$$

2.2.2. Cost of request/reply model

Cost for each request and reply is assumed to c_{rr}. Thus total cost is $n\, c_{\text{rr}}$, where n is the number of client. When request rate is β, cost per time unit is:

$$\beta n c_{\text{rr}}.$$

Time delay for access after client's intention is t_{rr} as we assume. Time delay between event occurrence and recognition of client is depends on request rate (similar to polling rate):

$$\frac{1}{2\beta}.$$

Deadline meet ratio from client's intention is as follows: when $D \geq t_{\text{rr}}$, client can access data within the deadline; however, when $D < t_{\text{rr}}$, client cannot access data within the deadline.

Now, deadline meet ratio from the occurrence of event is analyzed. Client can access data within deadline when the client requests data within D after the occurrence of event. As client's request rate is β, deadline meet ratio is:

$$\int_0^D \beta \varepsilon^{-\beta t} \mathrm{d}t = 1 - \varepsilon^{-\beta D}.$$

2.2.3. Periodic (polling) model

Periodic model is appropriate for applications in which delayed message is acceptable. Cost of periodic model (periodic publish or polling) per period is $c_{\text{poll}}(\alpha, T) + c_{\text{delay}}(\alpha, T)$. Thus, cost per time unit is:

$$(c_{\text{poll}}(\alpha, T) + c_{\text{delay}}(\alpha, T))/T,$$

where $c_{\text{poll}}(\alpha, T)$ can be between c_{rr} and $\alpha T c_{\text{rr}}$.
If we assume periodic publish, cost per time unit is:

$$(c_{\text{pub}}(\alpha, T) + ns(n) c_{\text{sub}}(\alpha, T) + c_{\text{delay}}(\alpha, T))/T,$$

where $c_{\text{pub}}(\alpha, T)$ is between c_{pub} and $\alpha T c_{\text{pub}}$, $c_{\text{pub}}(\alpha, T)$ and $c_{\text{sub}}(\alpha, T)$ is be between c_{sub} and $\alpha T c_{\text{sub}}$, and $c_{\text{delay}}(\alpha, T)$ is proportional to between c_{delay} and $\alpha T c_{\text{delay}}$. Average time delay for access after client's intention is $T/2$. Time delay between event occurrence and recognition of subscriber is $T/2$.

Client can always access data within the deadline when $D \geq T$. When $D < T$, however, client can access data within the deadline of probability D/T. Client can access data within the deadline when it requests data after which the first following polling occurs within D during the polling period T. We assume that data access is evenly distributed during polling period T.

Now, deadline meet ratio from the occurrence of event is analyzed. Client can always access data within the deadline when $D \geq T$. When $D < T$, however, client can access data within the deadline of probability D/T, which is similar to the analysis of deadline meet ratio from the intention (Table 2).

3. Performance comparisons

We have conducted performance comparisons on simulated condition and verify the parameter values by empirical experiments. They are explained in the following subsections respectively.

3.1. Parametric analysis

In this section, we describe performance comparisons by parametric analysis. Table 3 shows system parameters we set for the comparisons. We verify cost parameters through empirical experiments and the results are shown in Section 3.2. Fig. 1 shows performance comparisons between publish/subscribe, request/reply, and polling systems. In this experiment, cost is communication cost for each transaction. Since publish/subscriber system disseminates data via server instead of individually for each client, it requires less cost than request/reply system. As the number of client node increases, the cost gap between two systems increases. Periodic polling system saves cost by transferring data once per period when delay cost is negligible. However, cost increases as delay cost increase. Polling system is viable approach for applications where data delay is allowed and delay cost is negligible.

Table 3
System parameters for analysis.

Parameters	Values
α (publish rate)	0.5
β (request rate or access rate)	0.5
c_{ps} (publish/subscribe cost per event)	2
c_{pub} (publish cost per event)	1
c_{sub} (subscribe cost per event)	1
c_{rr} (cost per request and reply)	2
$c_{poll}(\alpha, T)$ (cost of periodic publish)	1 or α T
$c_{delay}(\alpha, T)$ (cost of delaying publish)	0, T, or α T
$s(n)$ (effect of sharing among n subscribers)	$1/n - 1$
t_{ps} (time delay for publish/subscribe)	1
t_{proc} (processing time for request/reply)	1 or 5
t_{rr} (time delay for request and reply)	1
$t_{poll}(\alpha, T)$ (time for periodic publish)	1, T, or α T
D (relative deadline from user's access intention or event occurrence)	Variable

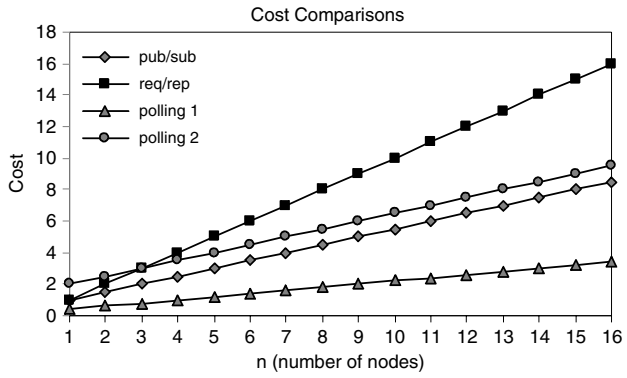

Fig. 1. Communication cost per transaction by varying number of clients (α = 0.5, $s(n)$ = 1, c_{ps} = 2, and c_{rr} = 2; $c_{pub}(\alpha, T)$ = c_{pub}, $c_{sub}(\alpha, T)$ = c_{sub}, and $c_{delay}(\alpha, T)$ = 0 for polling 1; $c_{pub}(\alpha, T)$ = $\alpha T c_{pub}$, $c_{sub}(\alpha, T)$ = $\alpha T c_{sub}$, $c_{delay}(\alpha, T) = 2\alpha T c_{delay}$ for polling 2).

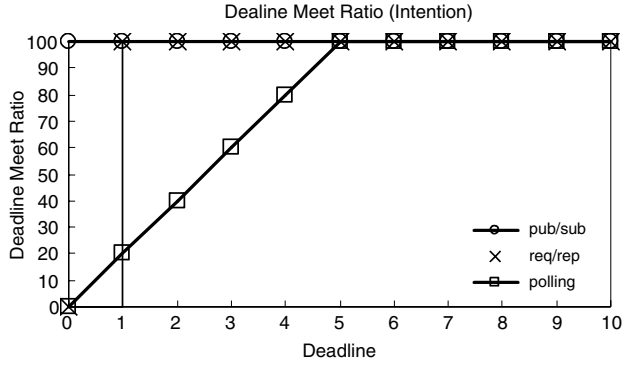

Fig. 2. Deadline meet ratio by varying deadline from user's access intention (t_{rr} = 1, t_{ps} = 1, T = 5, and β = 0.2).

Figs. 2 and 3 show deadline meet ratios between publish/subscribe, request/reply, and polling systems for user's access intention for pull based publish/subscribe model and for event occurrence respectively. As analyzed in the Section 2, we see the pub/subcurve meet the deadline better than req/rep and polling.

3.2. Experimental results

To verify the simulated result, we conducted empirical experiment using embedded system clients and a message brokering system. The purpose of our experiment was to get actual $c_{ps}(t_{ps})$ and $c_{rr}(t_{rr})$ which are publish/subscribe cost (i.e. time delay) per event and request and reply cost (i.e. time delay), respectively, for

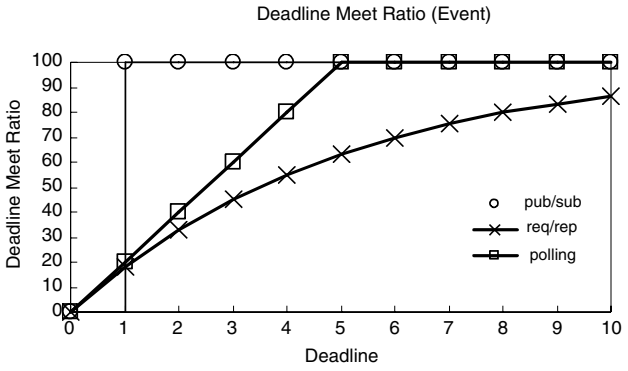

Fig. 3. Deadline meet ratio by varying deadline from event occurrence (t_{rr} = 1, t_{ps} = 1, T = 5, and β = 0.2).

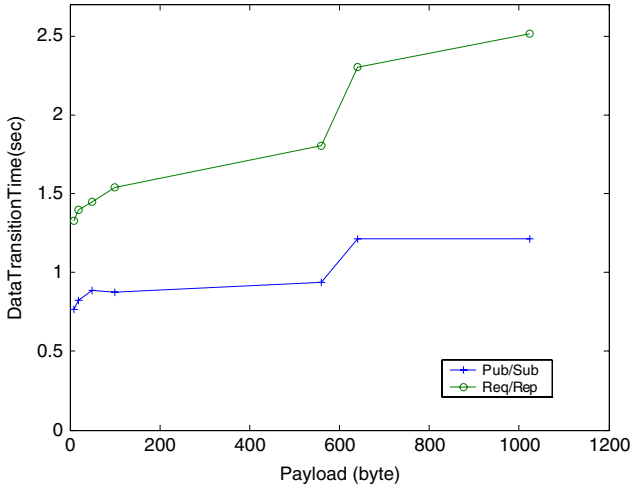

Fig. 4. Delay time by payload.

both different message sizes and numbers of clients in a practical environment. The experiment environment consists of NaradaBrokering system which is a message brokering system with HHMS *(Held Message Service)* [20] Proxy plug-in for mobile and embedded client. NaradaBrokering is developed at the Community Grids Laboratory at Indiana University. It is a content distribution infrastructure which supports asynchronous publish/subscribe communication model and originally designed for a uniform software multicast to support a real-time collaboration. We choose to use HHMS for the experiments because mobile or embedded devices are popular choice of client in a distributed real-time system.

We performed two types of experiments. First is the experiment to measure the data transition time between an event source (publisher) and an event displayer (subscriber) by varying the size of message (i.e. size of payload). We performed on the wireless environment which is a common network environment for a distributed real-time system such as a military system on the field. Since correct measurement of data transition time on the embedded device is not an easy task to achieve, we measured a round trip time (RTT) on the event source and get c_{ps} = RTT where $c_{ps} = c_{pub} + c_{sub}$. A client application (i.e. subscriber) on Treo 600 mobile phone device [21] which is connected to Internet through 2nd generation CDMA service just echoes back message from the event source (i.e. publisher) which runs on Linux machine. We did the same to get c_{rr}, 'Cost of request/reply event'.

The experiment result of the data transition time of publish/subscribe message (t_{ps}) and the data transition time of request/reply message (t_{rr}) is shown in Fig. 4. From the graph, we can get the relationship between t_{ps} and t_{rr}.

$$t_{rr} = t_{ps} + k, \tag{1}$$

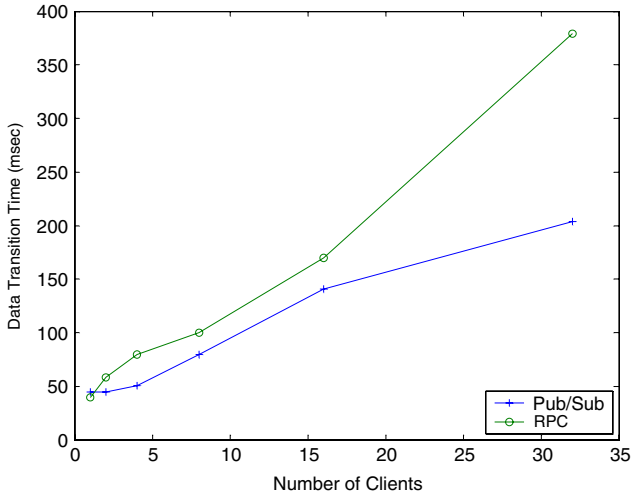

Fig. 5. Delay time by a number of clients.

where k is a constant. The k is relatively small to t_{ps} and t_{rr} if we increase the size of the message (i.e. payload). Thus, our system parameter setting in Section 3.2, $c_{ps} = c_{rr}$ and $t_{ps} = t_r$ are valid.

The second experiment is to measure the communication cost per transaction for varying number of clients. Conducting an experiment with a large number of clients is not acceptable in many cases and we were in the same situation where we have a limited number of mobile embedded devices. Thus, we performed the experiments using J2ME simulators. It is not a quite similar experimental environment compare to "simulation result" in Section 3.1. The simulator in this experiment is a software platform where the actual application runs on, thus it is more like to make an application run on a virtual device. The experimental result is shown in Fig. 5. From the result, we can see that the curves on both Figs. 1 and 5 are much resembled. Using these verified parameters, we perform simulations for comparing real-time performance (deadline meet ratio). Thus, we verify our simulation result.

The experiment is performed on a Linux machine equipped with Pentium III 1 GHz CPU and 512 MB memory and a mobile embedded device, Treo 600 equipped with 33 MHz Motorola Dragonball processor and 8 MB of memory. Time is measured with the Linux native timer by JNI. The subscriber application on mobile embedded device is written in Java Micro Edition for embedded and mobile device with MIDP 2.0.

4. Conclusion

Although publish/subscribe system has been popular recently in distributed real-time systems and embedded systems (e.g. aircraft control systems, military applications, medical imaging systems), cost analysis model has not been suggested and verified yet. This is important because application developers can estimate the performance of their application by applying the theoretical model and its verification through simulations and empirical experiments. In this paper, we present our cost analysis model for publish/subscribe systems especially in distributed real-time system domains. The empirical result from our test bed verifies our cost model. By providing the simulation result and the empirical result which is based on our cost analysis model, we give theoretical proof to the known claim, the publish/subscribe system is well suited for the distributed real-time system.

Acknowledgement

This work has partially been supported by the 2007 new faculty research fund of Ajou University, the Ubiquitous Computing and Network (UCN) Project, Knowledge and Economy Frontier R & D Program of the Ministry of Knowledge and Economy (MKE), and by the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2009-C1090-0902-0003).

References

[1] P. Eugster, P. Felber, R. Guerraoui, A. Kermarrec, The many faces of publish/subscribe, ACM Computing Surveys 35 (2003) 114–131.
[2] R. Rajkumar, M. Gagliardi, L. Sha, The real-time publisher/subscriber interprocess communication model for distributed real-time systems: Design and implementation, in: Proceedings of the 1st IEEE Real-Time Technology and Application Symposium, May 1995, pp. 66–76.
[3] J. Kaiser, M. Mock, Implementing the real-time publish/subscriber model on the Controller Area Network (CAN), in: Proceedings of the 2nd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, May 1999, pp. 172–182.
[4] Y. Huang, H. Garcia-Molina, Publish/subscribe in a mobile environment, Wireless Networks 10 (2004) 643–652.
[5] B.Y. Zhao, H. Ling, J. Stribling, S.C. Rhea, A.D. Joseph, J.D. Kubiatowicz, Tapestry: A resilient global-scale overlay for service deployment, IEEE Journal on Selected Areas in Communications 22 (2004) 41–53.
[6] S. Pallickara, G.C. Fox, NaradaBrokering: A middleware framework and architecture for enabling durable peer-to-peer grids, in: Proceedings of ACM/IFIP/USENIX International Middleware Conference Middleware2003, Rio de Janeiro, Brazil, June 2003.
[7] Antony I.T. Rowstron, Anne-Marie Kermarrec, Miguel Castro, Peter Druschel, SCRIBE: The design of a large-scale event notification infrastructure, in: Networked Group Communication, 2001, pp. 30–43.
[8] M. Ripeanu, Peer-to-Peer architecture case study: Gnutella network, in: First International Conference on Peer-to-Peer Computing, P2P'01, 2001.
[9] N.S. Good, A. Krekelberg, Usability and privacy: A study of Kazaa P2P file-sharing, in: Proceedings of the SIGCHI Conference on Human Factors in, 2003.
[10] G. Deng, M. Xiong, A. Gokhale, G. Edwards, Evaluating real-time publish/subscribe service integration approaches in QoS-enabled component middleware, in: Proceedings of the 10th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, ISORC, May 2007.
[11] P. Costa, M. Migliavacca, G. Picco, G. Cugola, Epidemic algorithms for reliable content-based Publish-subscribe: An evaluation, in: Proceedings of the 24th International Conference on Distributed Computing Systems, ICDCS04, 2004.
[12] J. Pereira, F. Fabret, F. Llirbat, D. Shasha, Efficient matching for web-based publish/subscribe systems, in: Proceedings of the 7th International Conference on Cooperative Information Systems, 2000, pp. 162–173.
[13] V. Ramasubramanian, R. Peterson, E.G. Sirer, Corona: A high performance publish-subscribe system for the world wide web, in: Proceedings of Networked System Design and Implementation, NSDI, San Jose, CA, May 2006.
[14] M. Caporuscio, A. Carzaniga, A. Wolf, Design and evaluation of a support service for mobile, wireless publish/subscribe applications, IEEE Transactions on Software Engineering 29 (12) (2003) 1059–1071.
[15] D. Carzaniga, A. Rosenblum,Wolf, Design and evaluation of a wide-area event notification service, ACM Transactions on Computer Systems 19 (2001) 332–382.
[16] M. Aguilera, R. Strom, D. Sturman, M. Astley, T. Chandra, Matching events in a content-based subscription system, in: Proceedings of the 18th Annual ACM Symposium on Principles of Distributed Computing, May 1999, pp. 53–61.
[17] G. Cugola, E. Di Nitto, A. Fuggetta, The JEDI event-based infrastructure and its application to the development of the OPSS WFMS, IEEE Transactions on Software Engineering 27 (9) (2001) 827–850.
[18] L. Fiege, G. Muhl, F. Gartner, A modular approach to building event-based systems, in: Proceedings of the 2002 ACM Symposium on Applied Computing, 2002, pp. 385–392.
[19] G. Pardo-Castellote, OMG data-distribution service: Architectural overview, in: Proceedings of the 23rd International Conference on Distributed Computing Systems, May 2003, pp. 200–206.
[20] S. Oh, G.C. Fox, S. Ko, GMSME: An architecture for heterogeneous collaboration with mobile devices, in: Proceedings of the fifth IEEE/IFIP Conference on Mobile and Wireless Communications Networks, Singapore, October 2003.
[21] Palm, Treo 600. http://www.palm.com/us/products/smartphones/treo600/.
[22] B. Segall, D. Arnold, J. Boot, M. Henderson, T. Phelps, Content based routing with elvin4, in: Proceedings of AUUG2K, Canberra, Australia, June 2000.

Sangyoon Oh received M.S. in Computer Science from Syracuse University, USA and Ph.D. in Computer Science Department from Indiana University at Bloomington, USA. He is an assistant professor of Division of Information and Computer Engineering at Ajou University, South Korea. He previously held a research position at SK Telecom, South Korea. His main research interest is in the design and development of web based large scale software systems and he has published papers in the area of mobile software system, collaboration system, Web Service technology, Grid systems, and Service Oriented Architecture (SOA).

Jai-Hoon Kim received the B.S. degree in Control and Instrumentation Engineering, Seoul National University, Seoul, South Korea, in 1984, M.S. degree in Computer Science, Indiana University, USA, in 1993, and his Ph.D. degree in Computer Science, Texas A&M University, USA, in 1997. He is currently a professor of the Information and Communication department at Ajou University, South Korea. His research interests include distributed systems, real-time systems, and mobile computing.

Geoffrey Fox received a Ph.D. in Theoretical Physics from Cambridge University and is now a professor of Computer Science, Informatics, and Physics at Indiana University. He is the director of the Community Grids Laboratory of the Pervasive Technology Laboratories at Indiana University. He previously held positions at Caltech, Syracuse University and Florida State University. He has published over 550 papers in physics and computer science and been a major author on four books. Fox has worked in a variety of applied computer science fields with his work on computational physics evolving into contributions to parallel computing and now to Grid systems. He has worked on the computing issues in several application areas — currently focusing on Earthquake Science.