

Client Mobility in Rendezvous-Notify

Sasu Tarkoma
Helsinki Institute for Information
Technology
P.O. Box 9800,
FIN-02015 HUT, Finland
sasutarkoma@hiit.fi

Jaakko Kangasharju
Helsinki Institute for Information
Technology
P.O. Box 9800,
FIN-02015 HUT, Finland
jaakko.kangasharju@hiit.fi

Kimmo Raatikainen
Helsinki Institute for Information
Technology
P.O. Box 9800,
FIN-02015 HUT, Finland
kimmo.raatikainen@hiit.fi

ABSTRACT

Event-based computing is vital for the next generation mobile services and applications that need to meet user requirements irrespective of time and location. The event paradigm is a form of asynchronous one-to-many communication and allows clients to receive information that matches their interests through filtering. Event-based communication is a good candidate for mobile computing, because it is asynchronous and supports disconnected operation. However, user and terminal mobility present problems pertaining to synchronization and delivery that need to be solved. In this paper, we examine and analyze mobility in the Rendezvous-Notify architecture. This event-delivery architecture is based on two server roles: access servers that maintain subscription information and buffered events, and resolution servers that are responsible for event channels and routing events to access servers. Access to event channels is done using a rendezvous mechanism.

General Terms

Performance, Experimentation

Keywords

Distributed events, mobile computing, session handover

1. INTRODUCTION

Event-based computing is vital for the next generation mobile services and applications that need to meet user requirements irrespective of time and location [1,2,8,12]. The event paradigm is a form of asynchronous one-to-many communication and allows clients to receive information that matches their interests through filtering. However, user and terminal mobility present problems pertaining to synchronization and delivery that need to be solved [5,6].

This paper presents the Rendezvous-Notify architecture, which is a distributed event notification infrastructure especially for mobile computing. The key concepts in our architecture are event channel, session, access server, resolution server, and event

domain. Access to event channels is done using a rendezvous mechanism based on linear hashing. The architecture aims to meet the requirements of mobile users by supporting bounded delivery times and disconnected operation.

In Rendezvous-Notify there are two ways that can be used to facilitate user and terminal mobility within a domain. The first approach is to allow the sessions to be distributed across the event domain. This approach has extra messaging cost if the session is on some other server than the current access server. The second approach is to move the session data to the new server in a process called session relocation. This has extra cost during a handover because all user sessions are moved to the new server in a handover procedure. However, there are no additional costs when the client is stationary. We present a cost model for mobility in the proposed architecture, compare the two mobility mechanisms using simulation results and discuss their performance.

The rest of the paper is organized as follows. In Section 2 we present the Rendezvous-Notify and discuss the basic principles of mobility in this architecture. In Section 3 we analyze theoretical cost estimates for two architectural options. In Section 4 we give simulation results and discuss the performance of the two evaluated session distribution approaches.

1.1 Related work

Mobility is a relatively new issue in event-based computing. Only a few papers and projects have covered mobility in event systems [8,12]. The Siena notification service has been recently extended to support mobility [3]. Siena is a scalable architecture based on event routing. The extension is called Cometa, and it provides support for terminal mobility on top of a routed event infrastructure. The mobility extension has been verified using formal methods.

In addition to Siena, the JEDI framework supports mobile components. JEDI maintains causal ordering of events; however, the tree-topology is not scalable to high number of servers because the root node may become a bottleneck [6]. Elvin is an event system that supports disconnected operation using a proxy but does not support mobility between proxies [14]. Benchmarking of event systems has been discussed in [4].

Rendezvous mechanisms are used in IP multicast in the sparse mode Protocol Independent Multicast (RFC 2362), and peer-to-peer systems, such as SCRIBE [13].

2. Rendezvous-Notify

The Rendezvous-Notify is an overlay event architecture for mobile computing, which is based on two server roles: access

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DEBS '03, June 8, 2003, San Diego, California.

Copyright 2003 ACM 1-58113-000-0/00/0000...\$5.00.

servers that maintain subscription information and buffer events, and resolution servers that are responsible for event channels and routing events to access servers. Filtering is done in two phases: first on the resolution servers, and in the second phase on the access servers. Resolution servers contain a more generic set of filters, and the access servers maintain the full set of filters. We define the concept of an event session, which is an intermediary storage for events before the clients download them. Client subscriptions are grouped into sessions in order to facilitate multi-device operation and device mobility. Sessions consist of zero or more active or paused subscriptions and they are also generally long-lived and may be moved between access servers when clients relocate.

In order for the subscription topology to converge rapidly and to meet various non-functional requirements, such as total ordering of events, we partition the event world into domains. Each event domain is a separate administrative entity with its own set of servers. This creates two kinds of mobility: mobility within a domain, and mobility between domains. In this paper, we consider mobility within one event domain.

Access to event channels is done using a rendezvous mechanism. A distributed data structure based on linear hashing, such as LH* [10], is used to locate the event channel responsible for a given event type. This explicit-join rendezvous approach provides near constant event channel lookup times, and bounded operation in terms of application-level messaging hops within the event domain.

Figure 1 illustrates event notification in Rendezvous-Notify. First (1) a client publishes an event using the client-server protocol. The request message is processed at the server local to the client, and forwarded based on the access server routing table or lookup table. In the case of a distributed hash table, the event type is hashed and this hash-based identifier is used to lookup the server IP-address. The message is forwarded (2) to the resolution server responsible for that particular event channel. The responsible server evaluates the subscription information, which consists of the event type and filters, set of servers interested in receiving a notification and possible authentication information. A notification is multicast to interested servers (3), which then associate the notification with an active client session, and if possible, notify the client or buffer the notification according to the session configuration (4).

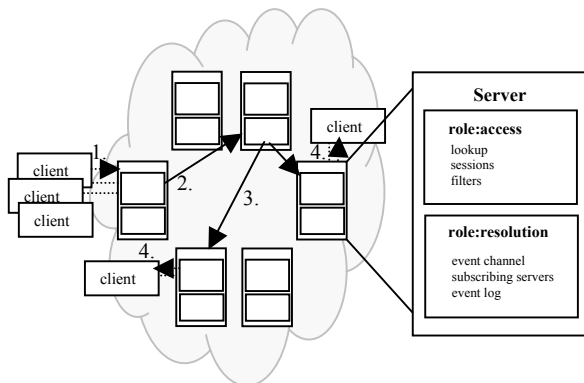


Figure 1. The Rendezvous-Notify architecture

From the mobility point, we need to consider terminal mobility and user mobility. Buffering incoming notifications for all device and user pairs supports user mobility. Applications on different terminals belonging to the same user may subscribe and receive notifications. Terminal mobility and roaming between access servers changes the dynamics of the system drastically.

We stress the importance of bounded subscription and event channel management operations cost both in terms of sent messages and in time. This is important for the system to support session exchange and ensure in reasonable timeframe that no events have been delayed or lost during mobility.

The proposed Rendezvous-Notify architecture meets this requirement by using a linear addressing mechanism for locating event channels responsible for a given event type. Implementations may support different algorithms for facilitating the lookup, including linear hashing, distributed hash table LH*, or a peer-to-peer algorithm similar to Scribe [13] or Tapestry [15]. By having a linear addressing space and using an explicit-join rendezvous scheme, instead of hop-by-hop routing modeled with a graph structure, the system supports constant subscription group management costs and scalability. Many current event systems use hop-by-hop event routing or flooding and may have a high subscription cost, which is not reasonable for mobility scenarios.

The proposed approach does not prevent connections between event channels; for example, event channels may be connected into a hierarchy, where published events are forwarded towards the root of the tree. In this case, the event publication cost may become greater; however, the lookup and subscription management cost for a single channel is not affected.

Figure 2 presents a high-level overview of the handover procedure in Rendezvous-Notify when session relocation is used. The event channel update operation and relocate operation have a cost of one operation and use RPC-semantics. Q denotes the size of the session that is transferred to the new server. Since the relocation request to the event channel (or channels) is done from the old server that still maintains the session, the client needs to know only the origin server of mobility.

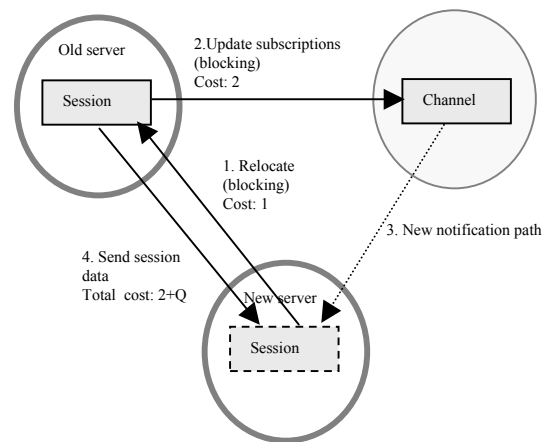


Figure 2. Handover procedure for session relocation in Rendezvous-Notify

We have verified this protocol with event buffering using Promela and Spin [7]. The handover protocol preserves order, does not

deadlock and does not lose events given that the underlying RPC architecture is reliable. The protocol is similar to the forward handoff procedure in the Wireless CORBA specification [11].

3. Evaluation of Architectural Options

In Rendezvous-Notify there are two architectural options: session distribution and session relocation. In the former approach, sessions are distributed and located on different access servers. In the analysis, we have assumed that a load-balancing algorithm uniformly distributes the sessions. The latter approach moves session data to a new server so that sessions always reside on the same server as the client. For the results in this paper, we assumed that the cost of an event channel lookup is one message. This parameter may vary depending on the distribution algorithm, which may be linear hashing, LH*, or a peer-to-peer routing algorithm.

Servers have two roles: access servers are responsible for sessions, that is, buffering events for mobile clients. Resolution servers are responsible for event channels and notifying subscribing clients by forwarding events to relevant access servers. Since our purpose is to evaluate mobility, the messaging cost is considered only from the viewpoint of the clients. Important random variables are mobility interval, mobility duration, publication interval, and the target server of mobility. The first three are modeled using the exponential distribution and the last random variable is modeled using the uniform distribution.

Below we derive the costs of event retrieval and publication for the two approaches. The cost metric is the number of external messages that are sent and received. The equations are derived from the four basic scenarios that are possible: both the event channel and session are on the current server, session is and channel is not, channel is and session is not, and finally both components are not on the current server (Figure 3). Session relocation has a messaging cost of 2 when the event channel resides on a different server, and 1 otherwise. Session distribution has a cost of 1 when both the channel and the session are on the current server, a cost of 2 when either of the two is on a different server, and a cost of 3 when the two components are on a different server. Here we have assumed that if both the session and channel are on a different node, a separate message is sent for the two components. The expected value for publication costs in the two scenarios is calculated by summing the probabilities of the four scenarios and their cost.

3.1 Event Retrieval Cost

Equation 1 defines the cost function of the event retrieval procedure, in which events are downloaded from the server, for a single client for session distribution given that all clients subscribe and produce only one type of event,

$$E(C_{SA}) = \frac{1}{M}(1+k) + (1-\frac{1}{M})(2(1+k)). \quad (1)$$

This illustrates how the messaging cost metric is structured. In the equation, M denotes the number of servers and k denotes the number of notifications that are downloaded or relocated. The $1/M$ gives the probability that the session is on the current access server. We calculate the expected cost in terms of messages. If the session is located on the current server, no further cost is incurred, and the client has sent one message and receives k

messages. If the session is on a different host, an extra message is sent to the server hosting the session, and k messages are sent to the current server in addition to the previous scenario. In essence, in the latter case the cost is doubled $(2*(1+k))$ ¹.

Equation 2 is the cost function for session relocation after mobility, where the parameter s represents the size of the relocated session queue, and $(1-1/M)$ represents the event channel update cost; the update cost is 1 if the channel is not located on the current server, and 0 otherwise:

$$E(C_{RA}) = \frac{1}{M}(1+k) + (1-\frac{1}{M})(1+k+s+(1-\frac{1}{M})). \quad (2)$$

3.2 Publication Cost

Equations 3 and 4 determine the expected publication cost of sending an event to the corresponding event channel for publication, for the session and relocation approaches respectively:

$$E(C_{SB}) = (\frac{1}{M})^2 + 4(\frac{1}{M}(1-\frac{1}{M})) + 3(1-\frac{1}{M})^2, \quad (3)$$

$$\lim_{M \rightarrow \infty} E(C_{SB}) = 3.$$

$$E(C_{RB}) = (\frac{1}{M})^2 + 3(\frac{1}{M}(1-\frac{1}{M})) + 2(1-\frac{1}{M})^2 = (1-\frac{1}{M}) + 1, \quad (4)$$

$$\lim_{M \rightarrow \infty} E(C_{RB}) = 2.$$

This cost does not include the messaging cost that incurs when the event channel forwards the notification to subscribing access servers. This latter cost is the same in both approaches and, therefore, it is not necessary in the comparison.

We have assumed that the publication procedure updates both the session and the event channel. If the session data does not need to be updated the publication estimate is the same for both approaches (Equation 4). In addition, if client mobility is not present, the publication cost is the same for both approaches (Equation 5):

$$E(C_{SB}) = E(C_{RB}) = \frac{1}{M} + 2(1-\frac{1}{M}). \quad (5)$$

In this case, there is no need to move sessions and approaches use either the first or the second scenario identified in figure 3. If mobility is present, eq. 4 has smaller coefficients than eq. 3. Looking at the publication cost alone, we can say that session relocation is more efficient; however, the situation changes when we take event retrieval also into account. Eq. 2 grows with the size of the session queue.

Event subscription and unsubscription has an impact on the performance of the models; however, the subscription and unsubscription cost functions in the two approaches are identical to the respective publication cost functions in equations 3 and 4, and we do not explicitly model subscription and unsubscription using separate parameters. Given the different combinations of session and channel locations, the subscription and unsubscription processes need to modify both session and channel status. We have four possible combinations for a given client and server

¹ We assume implicit ACKs or that NACKs are used.

(Figure 3). It follows that the subscription and unsubscription processes are identical to the publication process.

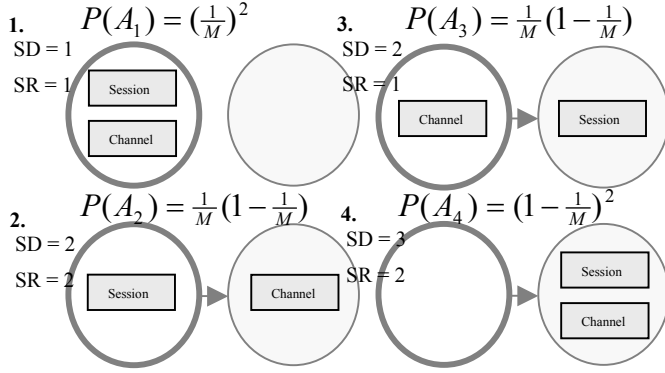


Figure 3. Four different configurations for event publication cost with mobility. SD denotes session distribution cost, SR session relocation cost

3.3 Estimates for Session Distribution and Relocation

Equation 6 presents the expected value S_d for event retrieval in session distribution, and equation 7 the estimate for event retrieval in session relocation S_r :

$$E(S_d) = n_c E(N_p) + E(N_m) \left[\frac{1}{M} + \left(1 - \frac{1}{M}\right) (E(Q) + 2 + E(N_f)) \right], \quad (6)$$

$$E(N_m) = \frac{T n_c}{\lambda_M + \lambda_d}, E(N_p) = \frac{T n_c}{\lambda_p}, E(N_f) = \frac{\lambda_M n_c}{\lambda_p},$$

$$E(S_r) = n_c E(N_p) + E(N_m) \left[\frac{1}{M} + \left(1 - \frac{1}{M}\right) (E(Q) + 3 - \frac{1}{M}) \right], \quad (7)$$

$$E(Q) = \frac{\lambda_d n_c}{\lambda_p}.$$

T denotes the simulation end time in minutes; n_c is the number of clients per server, λ_p denotes the mean publication interval, λ_M mean mobility interval, and λ_d the mean duration of mobility in minutes. N_m denotes total number of mobility occurrences; N_p is the number publications during time T , N_f is the expected number of event arrivals when a client is connected to a server, and Q is the expected number of events in a session after mobility.

In both equations 6 and 7, $n_c * E(N_p)$ denotes the total number of events that are downloaded during the simulation. Equation 6 takes also into account the number of events that are routed through the server responsible for the session. This expectation is calculated by multiplying the probability of using the server with the session and the cost of initiating the event download (1 message), and summing it with the complement probability multiplied with the cost of routing. The cost of routing is calculated by summing the expected queue size $E(Q)$ after mobility with the messaging cost (the download request message that is forwarded having cost of 2 messages) and the expected number of events that are routed during the time the client is

connected with the server. Equation 7 uses a similar strategy for calculating the expected cost of session relocation.

Note that since the simulation is finite and duration of mobility may limit the occurrence of departures, we use the expectation $\lambda_M + \lambda_d$ for the mobility interval. In addition, the download request message is sent only after mobility. Equation 8 presents the performance metric based on the absolute cost in both retrieval and publication:

$$C = E(S_r + C_{RB} N_p) / E(S_d + C_{SB} N_p). \quad (8)$$

3.4 Impact of Queue Size

The maximum queue size and event download rate are important parameters for the queue relocation cost function. Initially, we used infinite event download rate when the mobile nodes are connected. If the clients use slow wireless connections, their event download rate is limited. This prompts for mechanisms for managing the event queues.

In order to examine the impact of slow download rate, we also ran several experiments with different queue download rates and maximum queue sizes. The theoretical estimation of these models is more difficult.

3.5 Comparison of Session Relocation with Mobility in Event Routing

There are many ways to implement event routing and especially filtering in a routed environment. We have constructed a basic model for mobility in a routed environment in order to evaluate the performance of session relocation in Rendezvous-Notify.

The key metric is n ; the number of event routers between the source and destination of mobility. The cost function is different for subscription semantics and advertisement semantics. In subscription semantics, subscription messages are introduced at N servers, the number of nodes in the event network. In advertisement-based semantics only advertisement messages are introduced at every server.

The mobility protocol in event routing proceeds in four distinct phases: first, the target subscribes all events, then all n servers are pinged in order to ensure that the subscriptions have taken effect, the source unsubscribes, and finally the events are relocated and merged. In addition, there may be further costs triggered by changes in the subscription tables of the intermediate routers. The cost structure for this procedure is presented in table 1. The unsubscription cost depends also on other active subscriptions on the servers and is a worst-case estimate.

The advertisement-based approach is not reliable and should not be used, because if the target sends the advertisement for ping and publishes the ping event at the same time, the subscription from source has not taken effect yet and the event will be missed if buffering is not applied at intermediate servers or the notification is not included in the advertisement.

The approximate cost functions for one event type and client relocation for subscription semantics, advertisement semantics, and Rendezvous-Notify are as follows: $C_{RoutedSub} = 5N + 2n + Q$, $C_{RoutedAdv} = 2N + 4n + z + Q$, and $C_{Rendezvous-Notify} = Q + 2$. Q denotes the session size, and z the cost of filter subscription. We have assumed for the session relocation that the event channel is on some other server than the destination server and for the routed

costs that the changes in the subscription tables of intermediate nodes do not cause additional messaging, which may be the case if an advertisement is removed that has subscribers.

Table 1. Cost structure for mobility in event routing

Phase	Subscription semantics	Advertisement semantics
Source: Subsc. Ping(id)	N	0 (no advertisement yet)
Target: Subsc. Filter	N	z ($0 \leq z \leq N$)
Target: Subsc. Pong(id)	N	0 (no advertisement yet)
Target: Pub. Ping(id)	n	N (adv. + notif.)+n (subs.)
Source: Pub. Pong(id)	n	N (adv. + notif.)+n (subs.)
Source: Unsubscribe ² (ping, filters)	N	n
Target: Unsubscribe (pong)	N	n

The routed protocol is more complex than the two mobility mechanisms examined in this paper and it may not work reliably with advertisement semantics; however, the routed protocol works with various event systems based on hop-by-hop routing, and may support Internet-wide scalability. In the routed approach the cost function depends on the number of intermediate servers, the network topology, the queue size, and the processing time of the unsubscribe and subscribe operations. Rendezvous-Notify stores subscription information (filters) in sessions at access nodes, and therefore the client does not need to explicitly re-subscribe or know the current state of subscriptions.

4. Simulation Experiments

The goal of the discrete-event simulation study was to evaluate Rendezvous-Notify and session distribution from the mobility viewpoint, and the overall model was simplified for this purpose. The simulation model consists of clients and servers. Server communication has zero latency, and bandwidth is infinite. In the basic simulation scenario, there is only one event type that is subscribed by all clients, and all clients also publish events of that type. The client event download rate is infinite while they are connected to servers; queues build up during mobility. Each client has a single session, which are initially uniformly distributed over the servers in the simulation. The simulation was implemented using the C-language and the Simlib library [9].

Each simulation experiment was run with 5 replications, the simulation time was set to 400 hours. The simulation output was compared with the theoretical estimates given in the previous section. The performance index is the absolute cost of messaging in the relocation approach divided by the absolute cost of the session distribution approach.

4.1 Different Parameters

Figure 4 presents the performance index with various simulation parameters. This scenario has a variable number of servers and

² If target server subscribes events through the source server, the unsubscribe message is not propagated.

20 clients per server. In the figure, p denotes publication interarrival time, m = mobility interarrival time, d = mobility duration. Theoretical values are presented with points, and simulation output with points and 95% confidence intervals.

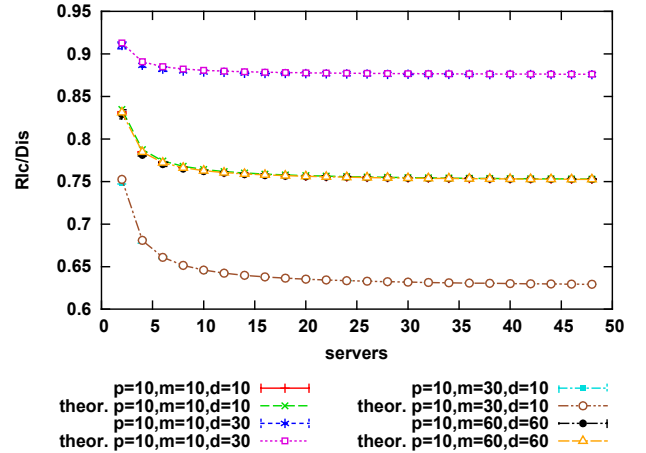


Figure 4. Experimentation with various parameters

4.2 Mobility

Figure 5 shows the impact of mobility and duration on the performance ratio of the two models. When the duration of mobility grows, the relocation approach has more events to move and the ratio grows. On the other hand, if the mobility interval grows, there are fewer relocations and the approach performs better.

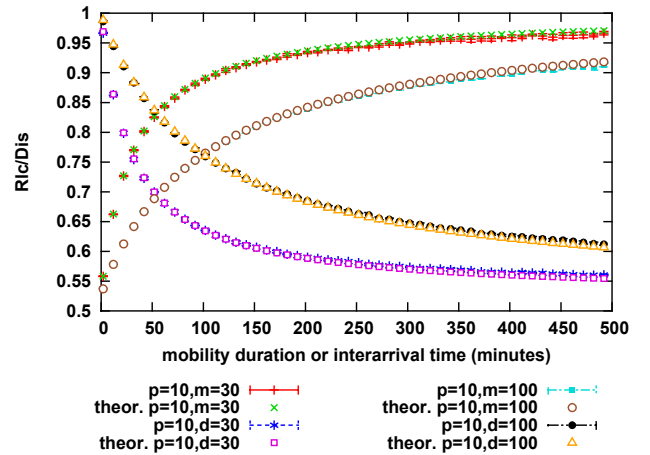


Figure 5. Impact of mobility

4.3 Queue Size and Download Rate

Figure 6 presents the impact of maximum queue size and varying download rate on the performance index with 10 servers and 20 clients per server. The publication interarrival time was 10, mobility interarrival time 80 and duration of mobility 200 minutes. As the queue size after mobility grows, the relocation approach performs worse than the session distribution approach. In addition, when the download rate is low, the session distribution has fewer messages to forward.

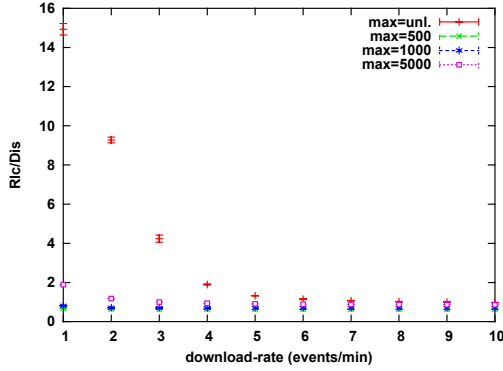


Figure 6. The impact of maximum queue size and download rate on the performance index

4.4 Factorial Design

In order to examine the impact of different parameters, we performed a 2^k factorial design experiment. We used the factorial design [9] to examine different parameters and measure interactions. Appendix A presents the factor-level combinations or design points. Table 2 presents the coding chart for the main effects of the study. Each design point was run for 200 hours of simulated time and with 5 replications.

The response results are presented in Appendix A. The metric is the performance index of the absolute cost of messaging in the two scenarios (relocation/distributed). The main effect of a factor is the average change in the response due to moving from “-“ level to the “+” level. Table 3 presents the values and their 95% confidence intervals for the main effects. The second-level effects were not statistically significant.

Table 2. Coding chart

Factor name	Description	-	+
Servers	Number of servers	5	50
Publication interval	Publication interval in minutes	Exp(5)	Exp(50)
Download rate	Maximum Number of events downloadable per minute.	Unlimited	20 Max queue size = 2000
Mobility interval	Mobility interval in minutes	Exp(10)	Exp(200)
Mobility duration	Mobility duration in minutes	Exp(10)	Exp(200)

Table 3. Main effects

Effect	Value
Servers	0.118583 ± 0.007117
Publication interval	-0.009727 ± 0.007661
Download rate	0.391024 ± 0.008868
Mobility interval	-0.593078 ± 0.010297
Mobility duration	0.590567 ± 0.008901

The performance ratio grows when the number of servers grows from 5 to 50. Change between the levels of the publication interval and mobility interval reduce the ratio. Change in the mobility interval is the largest effect with duration and results in a smaller ratio, because the number of relocations becomes smaller. From the responses in Appendix A, we can see that the performance ratio is over one in four responses when DL rate is limited, mobility is frequent, and durations are long, making the session relocation approach much better with this parameter space.

4.5 Discussion

In both theoretical and simulated results session relocation has consistently smaller absolute messaging cost than session distribution, given that the events accumulate only during mobility. When the client’s ability to download events is limited the queue sizes tend towards their maximum size and session distribution has lower absolute cost. The presence of wireless clients that have limitations on the download rate of events motivates the use of different queue management policies.

For systems that have a maximum queue size and queue management policies session relocation performs better than session distribution. However, this approach moves larger quantities of data at the end of the relocation, which places requirements on the data communication infrastructure used by the servers. In this study, we have not taken possible communication latencies into account. Session relocation has higher bandwidth usage variability than session distribution.

We compared the client mobility in Rendezvous-Notify with mobility in a generic routed event model. Rendezvous-Notify is less complex and has a smaller messaging cost function and more predictable completion time of the handover protocol.

5. Conclusions

This paper has examined the impact of mobility in the Rendezvous-Notify architecture and evaluated two different approaches for managing client sessions: session distribution and session relocation. We presented a theoretical cost model for these two approaches, and discussed the simulation results. Based on both simulation output and the theoretical results, session relocation performs better in environments where the queues do not build up. If the queues grow uncontrollably or to a maximum size, the relocation approach has more events to relocate, and performs worse in high-mobility scenarios. The presence of wireless clients that have limitations on the download rate of events motivates the use of different queue management policies. The session relocation approach was compared with a generic model that uses event routing. Based on this study, the proposed session relocation in Rendezvous-Notify approach is better in terms of messaging cost and the time needed to complete the handover procedure. We plan to continue examining the Rendezvous-Notify architecture in a mobile environment.

6. REFERENCES

- [1] Bacon, J., Moody, K., Hayton, R., et al. Generic Support for Distributed Applications. IEEE Computer, March 2000.
- [2] Caporuscio, M., Carzaniga, A., Wolf, A. An Experience in Evaluating Publish/Subscribe Services in a Wireless Network. In Third International Workshop on Software and Performance, Rome, Italy, July 2002.
- [3] Caporuscio, M., Inverardi, P., Pelliccione, P. Formal Analysis of Clients Mobility in the Siena Publish/Subscribe Middleware. Technical Report, Department of Computer Science, University of Colorado, October 2002.
- [4] Carzaniga, A., Wolf, A. A Benchmark Suite for Distributed Publish/Subscribe Systems. Technical Report. Department of Computer Science, University of Colorado, 2002. Available at: <http://www.cs.colorado.edu/~carzanig/papers/index.html>
- [5] Cugola, G., Di Nitto, E., Picco, G. Content-Based Dispatching in a Mobile Environment, 2000. In Workshop su Sistemi Distribuiti: Algoritmi, Architetture e Linguaggi (WSDAAL), 2000.
- [6] Cugola, G., Jacobsen, H.-A. Using publish/subscribe middleware for mobile systems. ACM SIGMOBILE Mobile Computing and Communications Review, Volume 6, Issue 4 (October 2002).
- [7] Holzmann, G. The Model Checker Spin, IEEE Trans. on Software Engineering, Vol. 23, No. 5, May 1997, pp. 279-295. Available at: <http://spinroot.com/spin/Doc/ieee97.pdf>
- [8] Huang Y., Garcia-Molina, H. Publish/Subscribe in a Mobile Environment. ACM Press. Second ACM international workshop on Data engineering for wireless and mobile access. Santa Barbara, California, United States, 2001.
- [9] Law, A. W. and W. D. Kelton. Simulation Modeling and Analysis (3rd ed.). New York: McGraw-Hill, Inc. 2000.
- [10] Litwin, W., Neimat, M., Schneider, D. LH* - Linear Hashing for Distributed Files. Hewlett-Packard Labs. 1993.
- [11] Object Management Group (OMG). Wireless Access & Terminal Mobility in CORBA, version 1.0, 2003.
- [12] Podnar, I., Hauswirth, M., Jazayeri, M. Mobile Push: Delivering Content to Mobile Users. In Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS'02), 2002.
- [13] Rowstron, A., Kermarrec, A., Castro, M., and Druschel, P. SCRIBE: The Design of a Large-scale Event Notification Infrastructure. Networked Group Communication, Lecture Notes in Computer Science, Vol. 2233, pp. 30-43, 2001.
- [14] Sutton, P., Arkins, R., Segall, B. Supporting Disconnectedness – Transparent Information Delivery in Mobile and Invisible Computing. CCGrid 2001 IEEE International Symposium on Cluster Computing and the Grid, Australia.
- [15] Zhao, B., Kubiawicz, J., and Joseph, A. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. U. C. Berkeley Technical Report UCB//CSD-01-1141, 2001.

Appendix A: Design matrix

Design Point	Servers	Pubs	DL Rate	Mobility	Duration	Response
1	-	-	-	-	-	0.775550 ± 0.001556
2	-	-	-	-	+	0.967108 ± 0.010629
3	-	-	-	+	-	0.581874 ± 0.001566
4	-	-	-	+	+	0.777847 ± 0.005240
5	-	-	+	-	-	0.783170 ± 0.000782
6	-	-	+	-	+	2.006898 ± 0.061124
7	-	-	+	+	-	0.582255 ± 0.001197
8	-	-	+	+	+	0.783201 ± 0.007873
9	-	+	-	-	-	0.788824 ± 0.001252
10	-	+	-	-	+	0.967660 ± 0.004554
11	-	+	-	+	-	0.584084 ± 0.001895
12	-	+	-	+	+	0.775251 ± 0.008666
13	-	+	+	-	-	0.794142 ± 0.001811
14	-	+	+	-	+	1.889393 ± 0.075864
15	-	+	+	+	-	0.586290 ± 0.004244
16	-	+	+	+	+	0.780576 ± 0.009555
17	+	-	-	-	-	0.752463 ± 0.000054
18	+	-	-	-	+	0.972373 ± 0.000530
19	+	-	-	+	-	0.533294 ± 0.000196
20	+	-	-	+	+	0.754420 ± 0.001484
21	+	-	+	-	-	0.765708 ± 0.000267
22	+	-	+	-	+	3.114041 ± 0.037634
23	+	-	+	+	-	0.533119 ± 0.000471
24	+	-	+	+	+	0.767283 ± 0.002536
25	+	+	-	-	-	0.754136 ± 0.000140
26	+	+	-	-	+	0.971953 ± 0.001164
27	+	+	-	+	-	0.533207 ± 0.000291
28	+	+	-	+	+	0.754553 ± 0.000721
29	+	+	+	-	-	0.766595 ± 0.000271
30	+	+	+	-	+	3.047394 ± 0.032089
31	+	+	+	+	-	0.533537 ± 0.000196
32	+	+	+	+	+	0.767373 ± 0.001300