Reliable Message Delivery for Mobile Agents: Push or Pull?

Jiannong Cao, *Member, IEEE*, Xinyu Feng, Jian Lu, Henry C. B. Chan, *Member, IEEE*, and Sajal K. Das, *Member, IEEE*

***Abstract*—Two of the fundamental issues in designing protocols for message passing between mobile agents (MAs) are tracking the migration of the target agent and forwarding messages to it. Even with an ideal fault-free network-transport mechanism, messages can be dropped during MA migration. Therefore, in order to provide reliable message delivery, protocols need to overcome message loss caused by asynchronous operations of agent migration and message forwarding. In this paper, two known message forwarding approaches, namely push and pull, are explored to design adaptive and reliable message delivery protocols. Based on a commonly used MA tracking model, the pros and cons of these two approaches are evaluated, both qualitatively and quantitatively. The comparative performance evaluation is presented in terms of network traffic and delay in message processing. We also propose improvements to the pull approach to reduce network traffic and the message delay. We conclude that with different message passing and migration patterns and varying requirements of real-time message processing, specific applications can select different message delivery approaches to achieve the desired level of performance and flexibility.**

***Index Terms*—Mobile agents (MAs), pull, push, reliable message delivery.**

I. Introduction

RECENT years have seen an explosion of interest in the mobile agent (MA) technology and its applications. MAs are autonomous objects or clusters of objects, which are able to move between locations in the so-called MA system. An MA system is a distributed abstraction layer that provides the concepts and mechanisms for mobility and communication on the one hand, and security of the underlying system on the other hand [1], [2]. MAs have a great potential for use in developing networking/distributed systems and a variety of applications including telecommunications, e-commerce, information searching, process coordination, mobile computing, and network management [3], [18]–[21].

MAs also provide a convenient and powerful paradigm for structuring distributed systems and applications. Agents can travel over the network to search for, filter, and process information required to accomplish their tasks. They can also cooperate with each other by sharing and exchanging information and partial results, and collectively making decisions. In various situations, MAs need to communicate with each other by passing messages [4], [5]. Remote interagent communication is thus a fundamental facility in MA systems. The abstractions and message passing mechanisms that form the basis for interagent communications can significantly impact the overall design and effectiveness of MA systems [17].

Although process communication has been a cliché in distributed systems research, the agent mobility raises a number of new challenges in designing message delivery mechanisms for effective and efficient communications between MAs. In particular, two fundamental issues must be addressed: 1) tracking the location of the target MA and 2) delivering the message to the agent. In the last several years, many MA/object tracking schemes have been proposed in different contexts, including mobile and wireless communication, and wide-area distributed systems. The common ground of these protocols has been that an intermediary is introduced to screen agents' mobility from message senders. For example, in the Mobile IP [6], the protocol designed for IP packets routing to mobile hosts, a mobile host registers its care-of-address with its home host, which then forward the IP packets to it. In some MA systems, such as Aglets [7] and Voyager [8], proxies are used to implement location transparency. Messages are sent to the proxy of the MA. The proxy keeps the current address of the agent and forward messages to it. In [9], a tracking agent is proposed for location tracking and message forwarding for cooperating agents. Several cooperating agents can share a tracking agent, which keeps their location information. The tracking agent forward incoming messages to the corresponding agents. In [10] and [11], we recently proposed a mailbox-based message delivery protocol for MAs. Messages are sent to mailboxes of their target agents. Agents query their mailboxes for messages whenever necessary. The mailbox can

Manuscript received December 5, 2002; revised January 26, 2004. This work was supported in part by the Hong Kong UGC's CERG Grant B-Q518 (PolyU 5076/01E), in part by the Hong Kong Polytechnic University's ICRG Grant A-PC53, in part by the China National 973 Program under Grant 2002CB312002 and 863 Program Grants 2001AA113110 and 2002AA116010, and in part by the US National Science Foundation under Grants IIS-0326505, EIA-0086260, and EIA-0115885. This paper was recommended by Associate Editor R. Rada.

J. Cao is with the Internet and Mobile Computing Laboratory, Department of Computing, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China (e-mail: csjcao@comp.polyu.edu.hk).

X. Feng is with the State Key Laboratory for Novel Software Technology, Department of Computer Science, Nanjing University, Nanjing 210093, China and also with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, China (e-mail: csxyfeng@comp.polyu.edu.hk).

J. Lu is with the State Key Laboratory for Novel Software Technology, Department of Computer Science, Nanjing University, Nanjing 210093, China (e-mail: lj@nju.edu.cn).

H. C. B. Chan is with the Internet and Mobile Computing Laboratory, Department of Computing, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China (e-mail: cshchan@comp.polyu.edu.hk).

S. K. Das is with the Center for Research in Wireless Mobility and Networking (CRewMan), Department of Computer Science and Engineering, The University of Texas, Arlington, TX 76019 USA (e-mail: das@cse.uta.edu).

Digital Object Identifier 10.1109/TSMCA.2004.826824

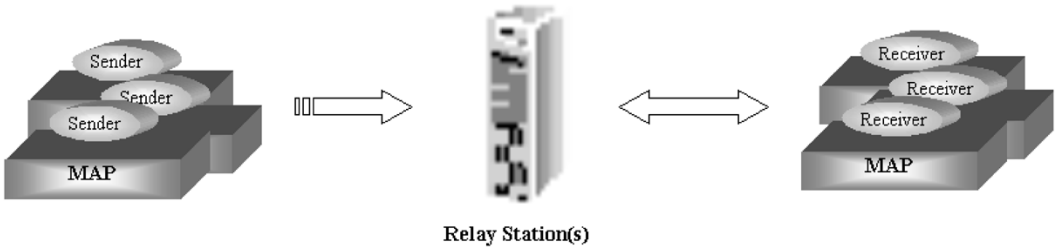


Fig. 1. Relay communication model.

be detached from its owner agent and can also migrate at a lower frequency, thus greatly reducing the cost of tracking the mailbox.

In the above existing protocols, intermediaries like the home server, the proxy, the tracking agent or the mailbox are used for agent tracking and message delivery. In this paper, this message-passing scheme is referred to as *relay communication*, which is illustrated in Fig. 1. Messages are passed between agents residing at different hosts, which are connected by the communication network. Each host runs a MA platform (MAP), providing communication, mobility, and security support for MAs. As shown in Fig. 1, there are three roles involved in the relay communication model, namely, *sender agents*, *relay stations*, and *receiver agents*. Each receiver agent in turn has one or more relay stations, which can be its home server, proxy, tracking agents or a mailbox. Communication between agents is divided into two steps: 1) the *transmission* of a message from the sender to the receiver's relay station and 2) the *delivery* of the message from the relay station to the receiver agent. To send messages, the message sender will first obtain the address of the target agent's relay station and then send messages to it. Later, the receiver agent can obtain messages from its relay station. Often the relay station is stationary and the message sender can obtain its address by resolving the receiver's ID [6]. In this case, message passing between the message sender and the relay station is easy to implement. Notice, however, the relay station itself can be mobile and there must be some way for the sender to track the relay station. This problem is not discussed in this paper. Interested readers are referred to [9] and [10] for details.

Protocols based on the relay communication model can effectively track the location of MAs and implement location-transparent communication, but additional efforts are needed to guarantee reliable delivery of messages to MAs. Here, by *reliability* we mean that no matter how frequently the target agent migrates, messages will be routed to it in a *bounded* number of hops without being dropped. Even with an ideal fault-free network transport mechanism, messages are not guaranteed to be reliably delivered to their destination MAs [12]. Because of the asynchronous nature of message forwarding and agent migration, when a message destined to an agent arrives at the current hosting site of the agent, the agent may have already left the host and is in migration to another site. The message has to be either discarded and resent later by the relay station, or be forwarded to the new location of the agent, as proposed in the forwarding pointer scheme. In both cases, the same situation may occur again and, as a result, the message may either finally get lost or keep chasing its target agent.

There exist two well-known approaches, namely, *push* and *pull*, for forwarding messages from the relay station to MAs. In the push mode, the relay station maintains the location of the MA and forward incoming messages to it. In the pull mode, on the other hand, the agent knows the address of its relay station and queries it periodically for messages. In this paper, we explore both of these two approaches for designing adaptive and reliable message delivery protocols. In particular, we investigate how to satisfy the reliability requirement under these two message-forwarding modes, discuss their pros and cons, and evaluate their performances by simulation experiments. The comparative performance evaluation is presented in terms of communication overheads and delay of message processing. Two improved versions of the basic pull approach, namely, *greedy pull* and *distance-based pull*, are also proposed to reduce network traffic and the message delay. We conclude that both the push and the pull modes can be made to guarantee message delivery between the relay station and the MA. For varying communication and migration patterns, specific applications can choose different message delivery modes to achieve the desired level of performance and flexibility.

The rest of the paper is organized as follows. Section II introduces the push and pulls modes of message delivery in the relay communication model. Their pros and cons are also analyzed qualitatively. In Section III, using simulations, we evaluate the performance of the push and pull approaches in terms of communication overhead and delay of message processing. Improvements to the basic pull mode are proposed in Section IV, while Section V further discusses the tradeoffs between push and pull modes. Finally, Section VI concludes the paper with the discussion of our future work.

II. Push and Pull in the Relay-Communication Model

In a more general context, push and pull are important concepts to describe the operation of distributed information dissemination. Informally speaking, if a user takes the initiative to request a specific piece of information, this is termed information *pull*. Otherwise, if the information supplier delivers information without the user's solicitation, the situation is characterized as information *push*. In this section, we investigate the push and pull approaches in the context of message passing between MAs using the relay communication model.

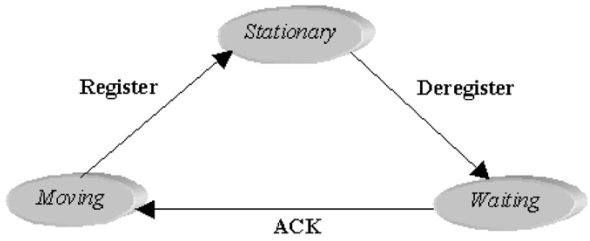


Fig. 2. State switching of an MA.

A. Push and Pull Approaches

As discussed above, push and pull are two possible modes in the relay communication model to forward messages from the relay station to the MA. In general, during the execution of an MA, there can be one or several relay stations serving the agent. To simplify the discussion, however, we assume that each MA is associated with only one relay station. This can be easily extended to situations where more than one relay station is used for forwarding messages to an agent.

Push: In the push mode, the relay station maintains the location information of the MA. Incoming messages destined to an agent are pushed to the current location of the target agent. Upon migration, after the MA reaches to the destination site, it registers the new location with its relay station. The relay station will update the agent's location information maintained in its database. Subsequent incoming messages destined to this agent are pushed to the agent's new address.

The simple push mode, although achieving location transparency, cannot guarantee reliable message delivery. It is possible for a message to be sent from the relay station toward the MA, and for the MA to move away before the message is delivered. That is, when a message is forwarded to the address as kept in the relay station, the target agent may have left for another host. Although it can be further forwarded, the message may keep chasing the target agent. To avoid message loss and the chasing problem caused by agent mobility, we propose a *synchronized push* mode. Synchronization between message forwarding from the relay station and agent migration is implemented in the following way. Before migration, the agent deregisters its current location with the relay station and waits for the ACK message. After it receives the ACK message, the agent migrates to the new location and registers its new location with the relay station upon arrival. As shown in Fig. 2, messages can be forwarded to the MA when it is in "stationary" and "waiting" states and must be blocked when it is in the "moving" state.

Since the agent will not move until it receives the ACK messages from the relay station, messages forwarded before the ACK message will have reached the target agent before its migration. No message will be forwarded during the migration of the target agent, i.e., during the interval between the ACK message and the next register message. Therefore, message loss and the chasing problem cannot occur [10].

Pull: In the pull mode, the relay station simply buffers the messages to the MA and does not need to keep its location information. The MA queries the relay station periodically for messages. Upon receiving a query message, the relay station forward the buffered message to the agent. If there is no message in the buffer, a "null" message is sent to the agent as a reply.

The MA can use either a synchronous or an asynchronous query operation. Synchronous query means the agent suspends its execution after issuing a query until it receives the reply from the relay station. In this way, the agent can ensure that no message will be forwarded to it during its migration. If asynchronous query is used, the agent can continue its execution after a query. However, to avoid message loss, the agent cannot migrate to other hosts until all the replies arrive.

The agent always knows the location of its relay station and initiates the request for messages, so location registration is unnecessary in the pull mode. Since the agent will not leave for the next host without receiving the response to its current query, there is no message loss and also the chasing problems cannot occur.

B. Properties of Push and Pull

In what follows, the properties of push and pull modes as well as their pros and cons are analyzed in terms of reliability, resiliency to failures, constraint on agent mobility, support for real-time processing, communication overhead, and flexibility.

1) Reliability: By reliability, we mean the messages can be routed to its target agent within a bounded number of hops. As discussed above, message loss and the chasing problem may occur under the simple push mode without synchronization. The synchronized push mode can avoid these problems and thus guarantee reliable message delivery [10]. In the rest of this paper, we only deal with the synchronized push mode, and the term push and synchronized push will be used interchangeably. In the pull mode, since the receiver agent takes the initiative to request messages from its relay station, the agent can ensure that no message will be forwarded to it during its migration. Therefore, the requirement of reliable message delivery can easily be satisfied.

2) Resiliency to Failures: In the push-based approach, the location and status (e.g., stationary, moving, and waiting as shown in Fig. 2) of the agent must be maintained at the relay station during the agent's life cycle. The state of the agent is lost if the relay station fails. After recovery, the relay station may have lost the trace of the agent. Moreover, the agent cannot detect the failure of the relay station and reregister with it until its next migration. In contrast, the pull-based relay station is resilient to failures due to its stateless nature. By periodically querying the relay station, the agent can easily detect failure of the relay station.

3) Constraint on the Agent Mobility: In the synchronized push mode, the agent has to deregister with its relay station and wait for the ACK message before its migration, therefore the agent mobility is constrained and the migration time is increased. In the pull mode, if synchronized query operation is used, the agent can leave for next host as soon as it finishes its execution at this host, but the execution time is increased. For asynchronous query, the agent also has to wait for the arrival of all the response to its query before migration. However, by

deciding the time and number of queries, the agent can flexibly reduce the constraint on its migration.

4) Support of Real Time Message Processing: In the push mode, unless the agent is in "moving" status, messages are forwarded to their target agents immediately after they arrive at the relay station. The sender has greater certainty that the message will reach its target within an appropriate timeframe. However, in the pull mode, the transmission time of a message depends not only on the network delay, but also on the frequency at which the receiver queries its relay station. Therefore, the delayed time for the message getting processed by the receiver is longer in the pull mode.

5) Communication Overheads: In the pull mode, two messages are needed for each query, namely, the query message and the response from the relay station. Moreover, to decrease the delay of message processing, the receiver may query at a higher frequency than the frequency of the message arrival at the relay station. Therefore, the pull-based approach is liable to impose a larger load on the network. On the other hand, three extra messages, namely, deregister, ACK and register, are needed for each agent migration in the push mode. In the cases where the agent migrates frequently but seldom communicates, the communication overhead of the push mode is significant.

6) Flexibility: Since the agent has the autonomy to decide on the time and frequency of the queries for messages, more flexibility is introduced in the pull mode. For example, the agent can adjust its query frequency dynamically. If it is in urgent need of information from its coordinator, it may query at a higher frequency. Otherwise, a lower frequency is adopted. Distance can be another factor of concern. If the current location of the agent is very far from its relay station, it can query the relay station at a much lower frequency or does not query at all. When it migrates to a host nearer to its relay station, it can query more frequently and process more messages buffered in the relay station.

III. Performance Analysis of Push and Pull

In this section, we analyze the performance of the push and pull mode in terms of *network traffic* and *delay of message processing*. Simulations are carried out to evaluate the performance. Before proceeding further, let us first define the following parameters.

n	Number of agent migrations.
t_m	Message interarrival time.
λ	Mean arrival rate of messages, i.e., the expected the number of messages that arrive within one unit of time.
t_r	Time that the MA resides in a host.
$1/\mu$	Expectation of the agent's residence time at a host.
η	Message to migration ratio, i.e., $\eta = \lambda/\mu$.
τ	Ratio of the query frequency of the agent to the arrival rate of the messages ($\tau > 1$).
$t_a(i)$	Time when message m_i arrives at the relay station.
$t_s(i)$	Time that message m_i is sent from the relay station to the target agent.

To compare the network traffic incurred by, respectively, the push and pull modes in the relay communication model, we only need to consider the communication cost between the agent and the relay station. This is because the costs of message transmission between senders and the relay station are the same for the two modes. In the push mode, the communication cost involves the transmission cost of deregister messages ($C_{\text{deregister}}$), ACK messages (C_{ack}), register messages (C_{register}), and all the agent messages (C_{msg}) that the relay station forward to the receiver. Hence, the total communication cost of the push mode during the agent's life cycle is given by

$$C_{\text{push}} = n(C_{\text{deregister}} + C_{\text{ack}} + C_{\text{register}}) + n\eta C_{\text{msg}}. \quad (1)$$

In the pull mode, the query frequency is proportional to the message arrival rate and is given by $\tau\lambda$. The communication cost involves the cost of query messages (C_{query}), agent messages (C_{msg}) sent by the relay station as responses, and "null" responses ($C_{\text{null-reply}}$). Thus, the total communication cost is given by

$$C_{\text{pull}} = n(\tau\eta C_{\text{query}} + (\tau - 1)\eta C_{\text{null-reply}} + \eta C_{\text{msg}}). \quad (2)$$

We only consider the communication between one agent and one relay station. However, our study can be extended to situations where there are l relay stations and each relay station serves for k agents. As long as assumptions like the message interarrival time and agent residence time are the same for all the agents, the model presented above can remain unchanged and the communication costs will be given by $l \times k$ times of C_{push} and C_{pull}.

In our simulation study, the communication cost is characterized by the number of messages sent, size of the messages, and the distance traveled by the messages. The interval (t_m) between message arrival at the relay station, and the residence time (t_r) the agent spends at a host, are exponentially distributed with the expectation of $1/\lambda$ and $1/\mu$, respectively. The transmission delay of messages and agents are proportional to their size and the distance traveled by them. The mobility model of the agent is shown in Fig. 3. The distance between the relay station and the agent is uniformly distributed over [0, 100] and the angle θ is uniformly distributed over $[0, 2\pi]$. Table I shows the assumptions and parameters used in our simulation experiments.

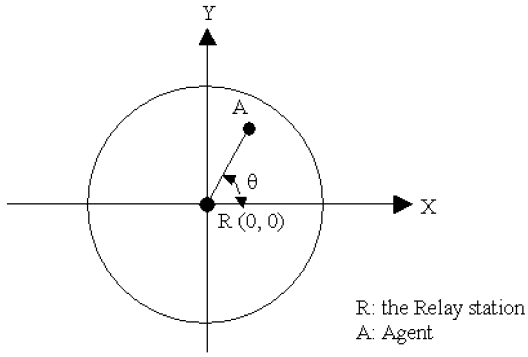


Fig. 3. Mobility and distance model used in our simulation.

TABLE I
ASSUMPTIONS AND PARAMETERS IN OUR SIMULATION

Parameters	Value	Justification
$C_{msg}(h_i \leftrightarrow h_j)$	$Distance(h_i, h_j)$	Communication cost of an agent message from host h_i to host h_j.
$C_{deregister}(h_i \leftrightarrow h_j)$, C_{ack}, $C_{register}$, C_{query}, $C_{null\text{-}reply}$	$Distance(h_i, h_j)/4$	Communication cost of a control message, which is smaller in size (¼ of that of the agent message).
$T_{msg}(h_i \leftrightarrow h_j)$	$Distance(h_i, h_j)$	Transmission time of a single agent message from host h_i to host h_j, which is proportional to the distance and message size
$T_{ctrl}(h_i \leftrightarrow h_j)$	$Distance(h_i, h_j)/4$	Transmission time of a single control message, e.g., register message, query message, etc.
$T_{agent}(h_i \leftrightarrow h_j)$ (in milliseconds)	$2 \times Distance(h_i, h_j)$	Transmission time of the agent, which is larger in size
$Distance(h_i, h_j)$	$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$	The distance between host h_i and host h_j, which is set to the geometric distance in the X-Y plane.
$1/\mu$	5 seconds	Mean residence time
n	100	The number of agent migrations

The simulation results with different values for message-to-mobility ratio (η) and query frequency (τ) are shown in Fig. 4. The communication costs for the push and pull modes (C_{push} and C_{pull}) are depicted in Fig. 4(a). We observe that when the agent migrates frequently but receives few messages, i.e., when the message-to-mobility ratio is small, the performance of the pull mode goes ahead of the push mode. This is because in the push mode, the agent has to register and deregister its location for every migration, no matter whether it will receive messages at the target host. However, when the number of messages received at each host increases the overhead of query messages in the pull mode outweighs that of the register and deregister messages in the push mode.

We use the average delay of message processing to evaluate the support of real time message processing. Suppose there are total of m messages forwarded from the relay station. Then, the average delay of message processing is given by

$$D = \sum_{i=1}^{m} t_s(i) - t_a(i)/m. \tag{3}$$

In our simulation, we assume that the workload of the relay station is very light and the query requirement of the agent is responded immediately after its arrival at the relay station. In the push mode, incoming agent messages are also processed by the relay station as soon as their arrival. Unless the target agent is in "moving" status, messages are forwarded to it without delay. Thus, our simulation result for the delay of message processing is the lower bound of the real value.

Fig. 4(b) illustrates the delay of message processing in the push and pull modes. We observe that the delay of message processing of the pull mode is much larger than that of the push mode.

IV. IMPROVEMENTS OF THE PULL-BASED APPROACH

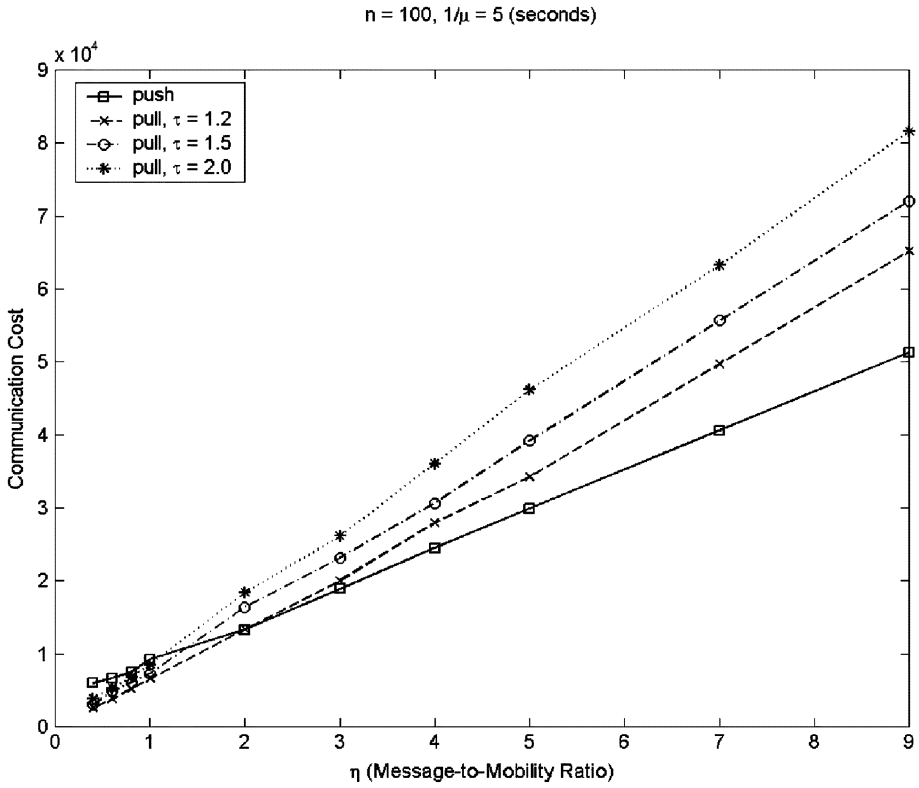
Simulation results presented in Section III show that the pull mode outperforms the push mode in terms of communication cost only when the message to mobility ratio is very low. Moreover, the pull mode will introduce much more delay in processing of messages by the receiver than the push mode. However, as we discussed in Section II-B, the pull mode can introduce more flexibility. Therefore, it is desirable if the message processing delay and communication overhead can be reduced. In this section, we propose two improvements to the pull-based approach, namely, greedy pull and distance based pull. To differentiate it from the proposed versions, the pull mode discussed in Section II is referred to as *simple pull* in the rest of this paper.

A. Greedy Pull

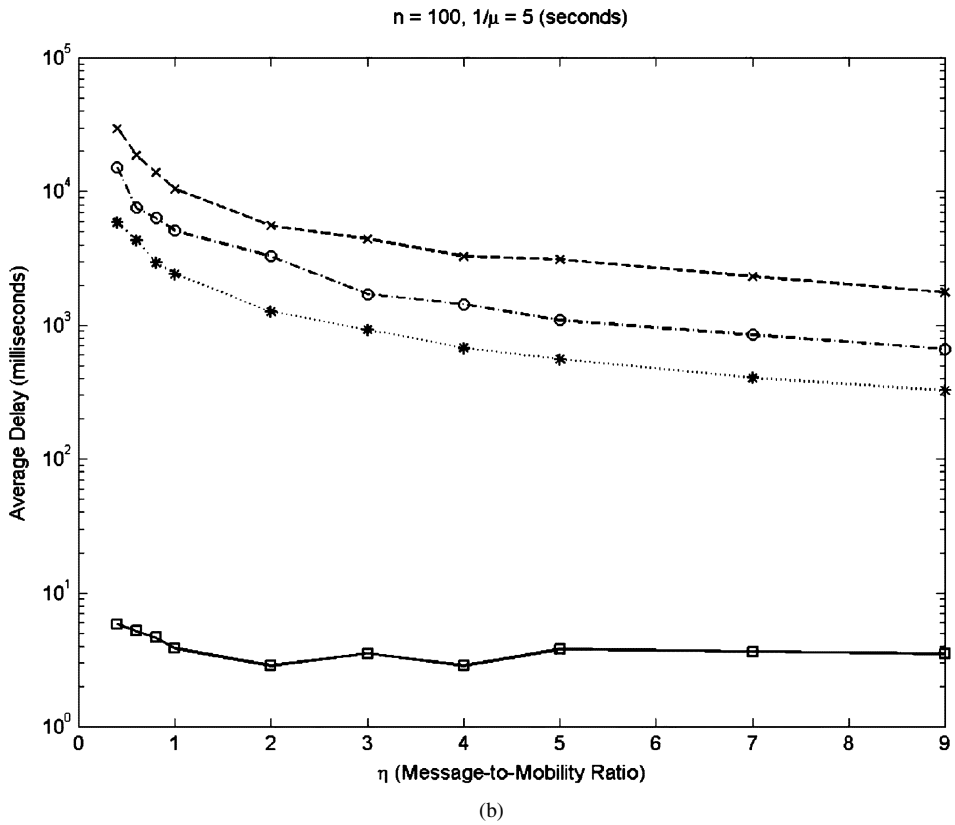
In the simple pull mode discussed, each time the agent queries the relay station for one message only. Even though there may be many messages in the buffer, the relay station forwards only one message as the response to one query from the agent. In this approach, the agent has more autonomy and flexibility to process the messages, since it can decide on the exact number of messages forwarded from the relay station. The constraint on the agent mobility is also lesser in this method because, for each response, at most one message is forwarded from the relay station. The agent does not need to wait for too long for the arrival of the response before its migration. However, if the agent needs to process all the messages buffered by the relay station, it has to query the relay station more frequently than necessary. To get all the incoming messages, the ratio of the query frequency to the message arrival rate, namely, τ, must be greater than 1. It is a waste of bandwidth and the delay of message processing is also large.

There is another way for the relay station to process the query messages from the agent, i.e., forwarding all the buffered messages, if any, in a batch to the agent as the response. We call this approach the *greedy pull* because the agent requires all the messages in one query. Greedy pull is actually a hybrid method of the push and pull modes because messages may be forwarded to the agent without explicit solicitation and beyond its expectation. In this mode the agent can query the relay station at a much lower frequency and the ratio τ can be much lower than 1 as long as the agent is not in urgent requirement of the messages.

Fig. 5(a) and (b) shows the performance comparison of the simple and the greedy pull modes. The assumptions and parameter setting of our simulation are the same as those in Section III, except that if m (≥ 1) messages are forwarded to the agent in a



(a)



(b)

Fig. 4. (a) Communication cost of the push and pull modes. (b) Delay of message processing under the push and pull modes.

batch, the communication cost is $m\times$ *distance* and the transmission delay is $m\times$ *distance* milliseconds. We can see that with the same query frequency, the communication cost of the greedy pull is very close to that of the simple pull mode. The former is a little larger, because more "*null*" responses are sent in the greedy pull mode. However, since the ratio τ can be lower

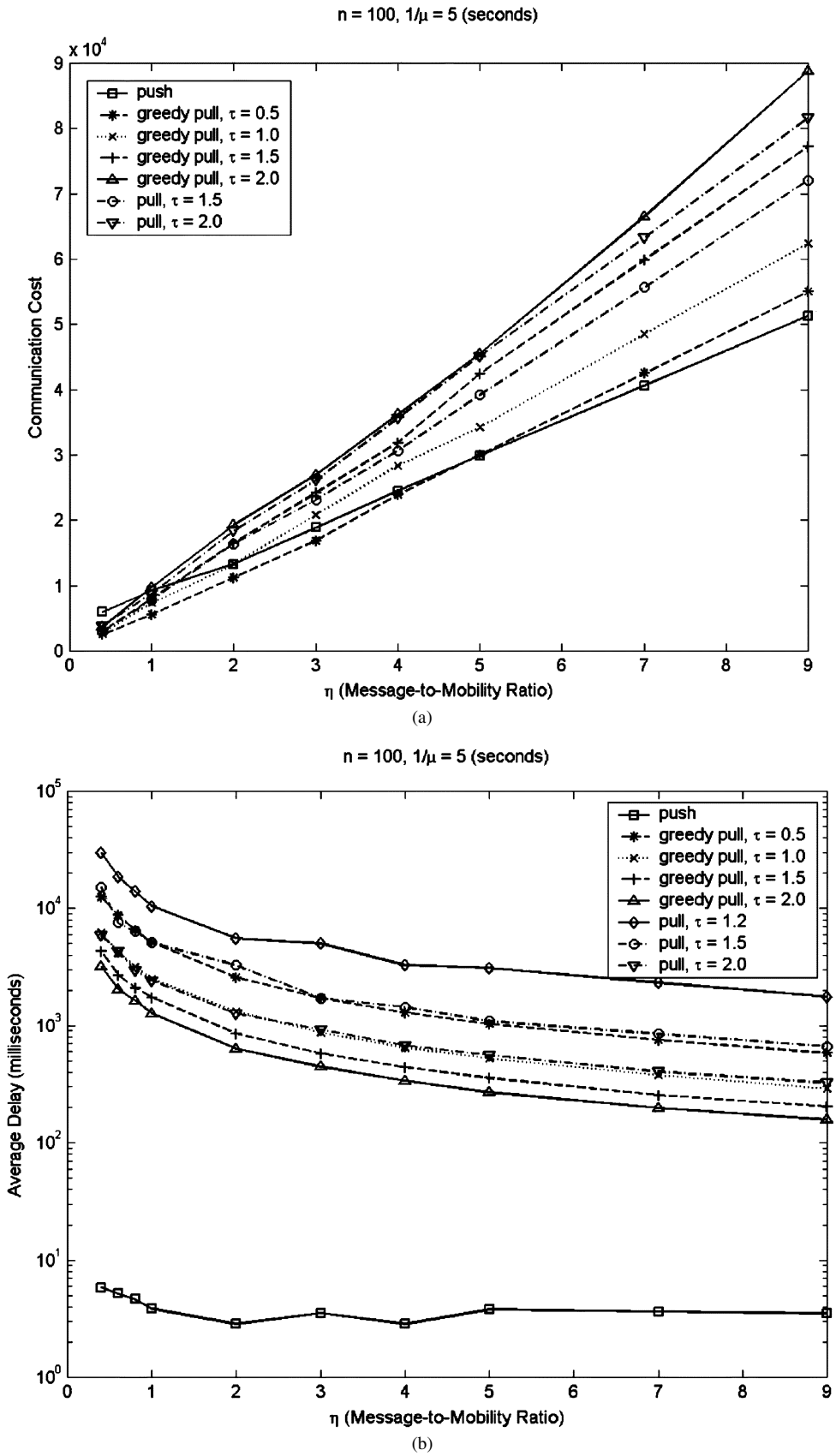


Fig. 5. (a) Comparison of the communication cost under the push, simple pull and greedy pull modes. (b) Comparison of delay of message processing under simple pull and greedy pull modes.

than 1, the communication cost of the greedy pull can be greatly reduced by using a lower query frequency. Moreover, with the same query frequency, the delay of message processing in the greedy pull mode is much lower. We can observe that in the

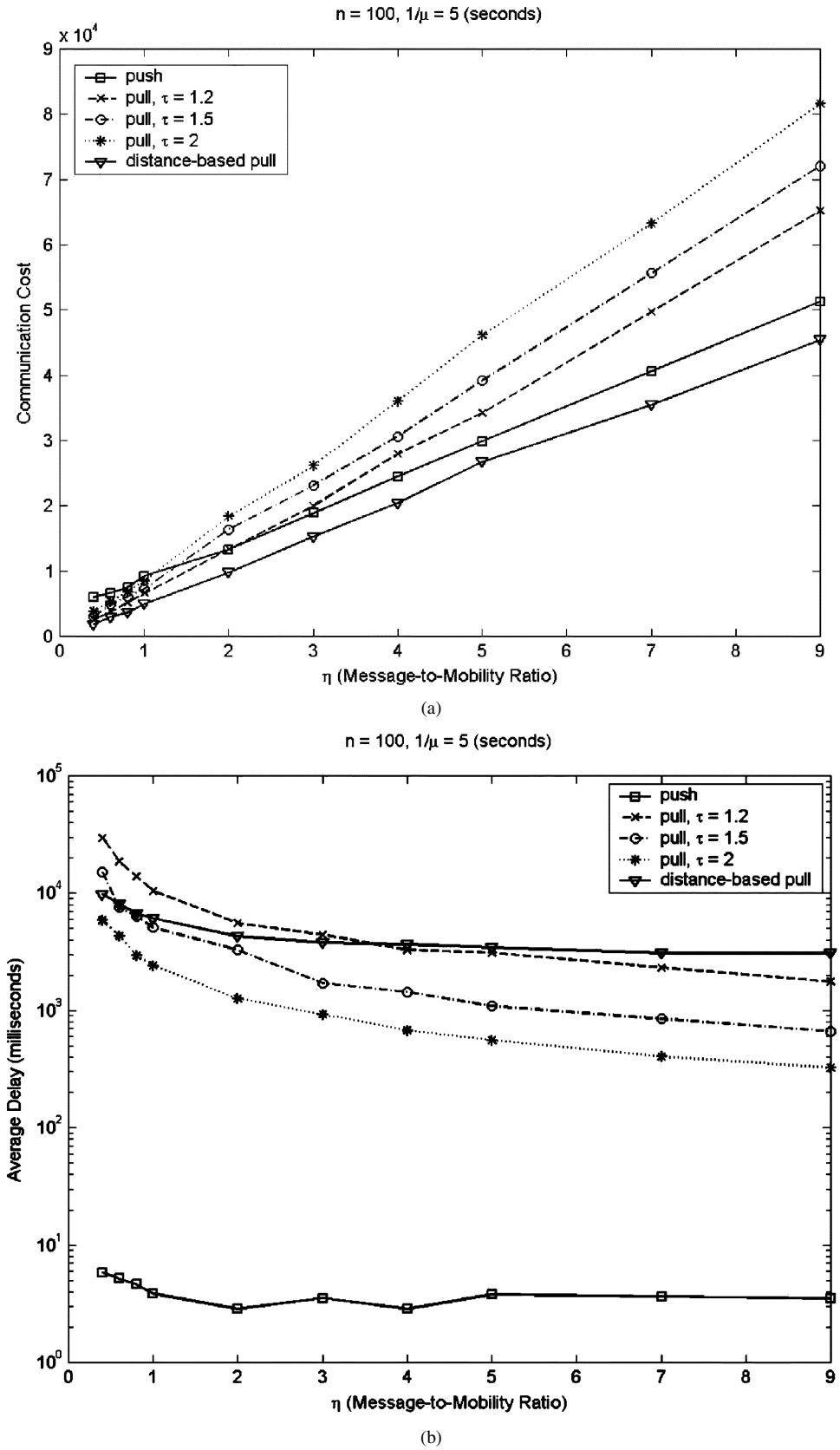


Fig. 6. (a) Comparison of the communication costs under the push, simple pull, and distance based pull modes. (b) Comparison of the delay of message processing under the push, simple pull, and distance based pull modes.

greedy pull mode, the average delay with the ratio of 0.5 and 1 are approximately the same as those of the simple pull mode with the ratio 1.5 and 2.0, respectively. Therefore, with the same tolerance for delay of message processing, the communication cost of the greedy pull mode is much lower than that of the simple pull mode.

TABLE II
PROPERTIES OF THE MESSAGE-DELIVERY MODES

	Reliability	Resiliency	Mobility constraint	Support of real time processing	Communication overhead	Flexibility
Sync Push	Yes	Low	High	Strong	Depends on the	Low
Pull	Yes	High	Low	Weak	Message-to-Mobility	High
Greedy Pull	Yes	High	Low	Medium	Ratio	Medium
Distance-based Pull	Yes	High	Low	Weak	Low	High

B. Distance-Based Pull

To better suit for different mobility patterns, many adaptive location update algorithms are proposed in the field of personal communication networks, including timer-based, movement-based, distance-based and state-based schemes [13], [14]. In these algorithms, mobile users decide whether to update their location information according to different merits. Like mobile users, the MA is also an autonomous object and can estimate the number of messages it will receive and decide whether and when to process them. The pull mode provides flexibility for the agent to make these decisions. The agent can query the relay station at various frequencies, which can be adjusted dynamically.

In this section, we propose an adaptive distance-based pull algorithm, in which the agent adjusts its query frequency based on the distance between it and the relay station as well as the message-to-mobility ratio. The term "distance" here is used to describe the number of hops between the agent's current residing site and the relay station. If the agent resides at a host far away from the relay station, it will reduce the query frequency, hence, less messages will be forwarded from the relay station. After it moves nearer to the relay station, the agent queries more frequently and processes more messages buffered at the relay station. To control the number of messages forwarded from the relay station, the agent requires one message in one query, as in the simple pull mode.

The agent-mobility and distance models in our simulation are the same as those defined in Section III. An empirical formula is used in our simulation to decide on the query frequency, which is given by

$$f = \tau\lambda = 2(1 - \mathrm{dist}(a, r)/\mathrm{mx\_dist})\lambda \tag{4}$$

where $\mathrm{dist}(a, r)$ is the estimated distance between the agent a and the relay station r; $\mathrm{mx\_dist}$ is the maximal possible value of the distances between any pair of agent and relay stations. mx_dist is set to 100 in our distance model. Thus, the ratio of the query frequency to the message arrival rate, denoted by τ, is totally determined by the distance and varies over [0, 2].

The communication cost of the distance-based pull algorithm is illustrated in Fig. 6(a). As can be seen, the communication cost is greatly reduced in the adaptive algorithm, which is even lower than the push-based algorithm. However, the performance improvement is at the cost of the additional delay and, therefore, the real-time support for message processing may be sacrificed. Since most messages are buffered at the relay station and processed only when the agent moved to a host near the relay station, the delay of message processing is increased, as shown in Fig. 6(b).

The receiver agent can adjust the query frequency according to its requirement of real-time message processing in addition to distance. If it is in urgent use of some information from its communication partner, the agent can increase the query frequency. Otherwise, a lower frequency can be used. Since the urgency requirement is very much application specific, it is not modeled in our simulation and users can take it into account in particular applications.

V. TRADEOFFS BETWEEN PUSH AND PULL

Push and pull are two canonical techniques for web data dissemination [15], [16]. In [15], it was shown that the push-based approach performs better than the pull mode in terms of the number of messages. However, it is not necessarily true in the mobile environment, where the receiver keeps moving and changing its address. In the push mode, since the MA has to register its address on arrival at a new host and synchronize the migration and message forwarding, the communication overhead is substantial if the message-to-mobility ratio is low. On the other hand, since the receiver has more autonomy in the pull mode to decide on the number of messages to be received from the relay station, more flexibility is introduced and adaptive algorithms can be designed to reduce the communication overhead.

From the results of our simulation experiments on the push- and pull-based algorithms and the two variants of the pull mode, we observe that the push mode is suitable for communication intensive applications, where the message-to-mobility ratio is high and the agent needs real time processing of messages. However, if the agent migrates frequently and smaller constraint on the agent mobility is preferred, the user can choose the pull mode. According to the specific requirement of applications, the greedy pull or distance-based pull can be chosen for lower delay of message processing or communication cost.

Table II summarizes our discussion and gives a comparison of the synchronized push, simple pull, greedy pull and distance-based pull approaches. Since the push and pull modes have complementary properties, applications with different requirements, such as fault resiliency, constraint on agent mobility, support of real time processing, communication overhead, and flexibility, can choose different communication modes.

For a better balance of the communication cost and the delay of message processing, a combination of the push and pull algorithm can be used. That is, the agent can switch between the pull and push modes. If currently a push mode is used and the agent wants to switch to the pull mode at the next host, it does not register its new address with the relay station after its arrival at the host. If messages are needed, the agent queries the relay station as in the pull mode. Without the register message, the status of the agent kept at the relay station is always "moving" and incoming messages are blocked at the relay station. Messages will not be forwarded to the agent unless the relay station

receives query messages or register message. If the agent wants to switch to the push mode, it just sends a register message to the relay station, which acts as a subscribe message. The relay station changes the status of the agent to "stationary" and resumes push of messages to the agent. The switch between push and pull can be decided by the number of messages the agent needs at the next host, the distance between the agent and the relay station, and the real-time message processing requirement.

VI. Conclusion

The relay communication model is widely used for MA tracking and message routing. In most of the existing algorithms, the push mode is used to forward messages to the agent. In [10] and [11], we proposed a mailbox-based algorithm and the agent pulls messages from the mailbox. In this paper, we abstract and identify the relay communication model from existing algorithms and explore the two possible approaches to implement reliable message delivery in this model, namely, synchronized push and pull. We show that the push and pull modes have complementary properties in terms of agent mobility constraint, communication overhead, support of real time message processing, relay station's resiliency to failures, and flexibility.

Our simulation results conclude that the push mode is suitable for communication-intensive applications, where the message-to-mobility ratio is high and the agent needs real-time processing of messages. However, if the agent migrates frequently and low constraint on the agent mobility is preferred, a user may choose the pull mode. According to the specific application requirements, greedy pull or distance-based pull can be chosen for lower delay of message processing or communication cost. For a better balance of communication cost and delay of message processing, urgency of the requirement of messages can be considered and a combination of the push and pull algorithms can be used in particular applications.

The simulation experiments were conducted under a uniformly distributed migration model. However, it is a challenging task to design the mobility model for MAs. Although several mobility models have been proposed for personal communication services (PCS) networks [14], the mobility model of the mobile user is quite different from that of the MA. Since the speed of user mobility is constrained by the speed of the transportation vehicles, the mobile user has to pass through adjacent cells in PCS networks during his migration and the mobility is restricted to a limited area within a fixed timeframe. However, it is quite different for the MA, which is a software object transmitted through the computer network. The migration pattern can be very different in different applications. The scope of migration also varies greatly from the local network to the Internet scope. In this paper, we used a uniformly distributed distance model to characterize the agent mobility. A refined model needs to be defined in specific applications.

References

[1] M. Straßer, J. Baumann, and F. Hohl, "Mole—A java based mobile agent system," in *Proc. 2nd ECOOP Workshop Mobile Object Syst.*, 1996, pp. 327–334.

[2] A. Pham and A. Karmouch, "Mobile software agents: An overview," *IEEE Commun. Mag.*, vol. 36, pp. 26–37, July 1998.

[3] D. B. Lange and M. Oshima, "Seven good reasons for mobile agents," *Commun. ACM*, vol. 42, no. 3, pp. 88–89, 1999.

[4] J. Cao, G. H. Chan, W. Jia, and T. Dillon, "Check pointing and rollback of wide-area distributed applications using mobile agents," presented at the Proc. IEEE Int. Parallel Distributed Process. Symp., San Francisco, CA, Apr. 2001.

[5] T. K. Shih, "Agent communication network—A mobile agent computation model for internet applications," in *Proc. IEEE Int. Symp. Comput. Commun.*, 1999, pp. 425–431.

[6] C. E. Perkins, "IP mobility support," presented at the RFC 2002, Oct. 1996.

[7] D. B. Lange and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets*. Reading, MA: Addison-Wesley, 1998.

[8] Objectspace Voyager Core Technology [Online]. Available: http://www.objectspace.com

[9] G. Kunito, Y. Okumura, K. Aizawa, and M. Hatori, "Tracking agent: A new way of communication in a multi-agent environment," in *Proc. IEEE 6th Int. Conf. Universal Personal Commun.*, vol. 2, 1997, pp. 903–907.

[10] X. Feng, J. Cao, J. Lu, and H. Chan, "An efficient mailbox-based algorithm for message delivery in mobile agent systems," in *Proc. 5th IEEE Int. Conf. Mobile Agents*, Atlanta, GA, Dec. 2001, pp. 135–151.

[11] J. Cao, X. Feng, J. Lu, and S. K. Das, "Mailbox-based scheme for mobile agent communications," *IEEE Comput.*, vol. 35, pp. 54–60, Sept. 2002.

[12] A. Murphy and G. P. Picco, "Reliable communication for highly mobile agents," *Agent Syst. Arch./Mobile Agents*, pp. 141–150, Oct. 1999.

[13] A. Bar-Noy, I. Kessler, and M. Sidi, "Mobile users: To update or not to update?," in *ACM/Baltzer J. Wireless Networks*, vol. 1, July 1995, pp. 175–195.

[14] V. Wong and V. Leung, "Location management for next generation personal communication networks," *IEEE Network*, vol. 14, pp. 18–24, Sept./Oct. 2000.

[15] P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy, "Adaptive push-pull: Disseminating dynamic web data," *WWW'2001*, pp. 265–274, May 2001.

[16] S. Acharya, M. J. Franklin, and S. B. Zdonik, "Balancing push and pull for data broadcast," in *Procs. ACM SIGMOD*, May 1997, pp. 183–194.

[17] M. Ranganathan, M. Bednarek, and D. Montgomery, "A reliable message delivery protocol for mobile agents," in *Proc. 2nd Int. Symp. Agent Syst. Applicat., 4th Int. Symp. Mobile Agents*, vol. 1882, 2000.

[18] Q. Hairong, S. Iyengar, and K. Chakrabarty, "Multiresolution data integration using mobile agents in distributed sensor networks," *IEEE Trans. Syst., Man, Cybern. C*, vol. 31, no. 3, pp. 383–391, Aug. 2001.

[19] H. Chan, H. Chen, T. Dillon, J. Cao, and R. Lee, "A mobile agent-based system for consumer-oriented e-commerce," in *Proc. 4th Int. Conf. Electron. Commerce*, Hong Kong, China, Oct. 23–25, 2002.

[20] P. Dasgupta, N. Narasimhan, L. E. Moser, and P. M. Melliar-Smith, "MAgNET: Mobile agents for networked electronic trading," *IEEE Trans. Knowledge Data Eng.*, vol. 11, pp. 509–525, July/Aug. 1999.

[21] N. Minar, K. H. Kramer, and P. Maes, "Cooperative mobile agents for dynamic network routing," *Software Agents Future Commun. Syst.*, 1999.

Jiannong Cao (M'93) received the B.Sc. degree in computer science from Nanjing University, Nanjing, China in 1982 and the M.Sc. and the Ph.D. degrees in computer science from Washington State University, Pullman, in 1986 and 1990 respectively.

He is currently an Associate Professor in the Department of Computing, Hong Kong Polytechnic University, Hung Hom, Hong Kong, China, where he is also the Director of the Internet and Mobile Computing Lab. He was on the Faculty of Computer Science at James Cook University and University of Adelaide, Australia, and the City University of Hong Kong. His research interests include parallel and distributed computing, networking, mobile computing, fault tolerance, and distributed software architecture and tools. He has published over 120 technical papers in the above areas.

Prof. Cao is a Member of the IEEE Computer Society, the IEEE Communication Society, and ACM. He is also a Memeber of the IEEE Technical Committee on Distributed Processing, IEEE Technical Committee on Parallel Processing, IEEE Technical Committee on Fault Tolerant Computing, and Computer Architecture Professional Committee of the China Computer Federation. He has served as a Member of Editorial Boards of several international journals, a reviewer for international journals and conference proceedings, and also as an organizing/program committee member for many international conferences.

Xinyu Feng received the B.Sc. and the M.Sc. degrees in computer science from Nanjing University, Nanjing, China, in 1999 and 2002, respectively.

He is currently a research assistant in the Department of Computing, the Hong Kong Polytechnic University, Hong Kong, China. His research interests include distributed systems and programming languages.

Jian Lu received the B.Sc., the M.Sc. and the Ph.D. degrees in computer science from Nanjing University, Nanjing, P.R. China, in 1982, 1984, and 1988, respectively.

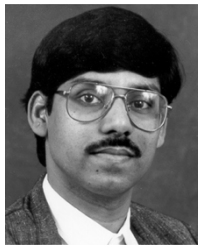
He is currently a Full Professor in the Department of Computer Science and Technology, Nanjing University. He is also the Director of the State Key Laboratory for Novel Software Technology and is the Vice Director of the Institute of Software Technology at Nanjing University. He has published over 90 technical papers in the above areas, which appeared in national and international journals and conference proceedings. His research interests include software automation, software agent, and middleware.

Dr. Lu is a Member of the Board of the UNU International Institute for Software Technology and ACM. He is also the Chairman of Systems Software Professional Committee of the China Computer Federation. He has served as an organizing/program committee member for many international conferences.

Henry C. B. Chan (S'95–A'97–M'98) received the B.A. and M.A. degrees from the University of Cambridge, Cambridge, UK and his Ph.D. degree from the University of British Columbia, Vancouver, BC, Canada.

From October 1988 to October 1993, he was with Hong Kong Telecommunications Limited, working primarily on the development of networking services in Hong Kong. Between October 1997 and August 1998, he worked with BC TEL Advanced Communications on the development of high-speed networking technologies and ATM-based services. Currently, he is an Assistant Professor in the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong, China. He has authored or co-authored a textbook on e-commerce, a book chapter in the Internet Encyclopedia, and more than 50 journal and conference papers. His research interests include networking and communications, wireless networks, electronic commerce (e-commerce), Internet technologies, and mobile computing.

Prof. Chan is a Member of the IEE and ACM. He is currently serving as an Executive Committee Member of the IEEE Hong Kong Section Computer Chapter. He has been listed in Marquis Who's Who in the World 2002.



Sajal K. Das (M'88) received the B.Tech. degree in computer science from the University of Calcutta, Calcutta, India, in 1983, the M.S. degree in computer science from the India Institute of Science,Bangalore, India, in 1984, and the Ph.D. degree in computer science from University of Central Florida, Orlando, in 1988.

He is a Professor of Computer Science and Engineering and also the Founding Director of the Center for Research in Wireless Mobility and Networking (CReWMaN) at the University of Texas, Arlington (UTA). He has published over 250 research papers, directed numerous funded projects, and holds 5 US patents in wireless mobile networks. His current research interests include resource and mobility management in wireless networks, mobile and pervasive computing, wireless multimedia and QoS provisioning, sensor networks, mobile Internet protocols, distributed processing and grid computing.

Prof. Das serves on the Editorial Boards of IEEE TRANSACTIONS ON MOBILE COMPUTING, *ACM/Kluwer Wireless Networks*, *Parallel Processing Letters*, and *Journal of Parallel Algorithms and Applications*. He served as the General Chair of IEEE PerCom'2004, ICNDS'2004, CIT'2003, and IEEE MASCOTS'2002; General Vice Chair of IEEE PerCom'2003, ACM MobiCom'2000, and HiPC'2000–01; General Chair of ACM WoWMoM'2000–02; Program Chair of IWDC'2002, WoWMoM 1998–99; TPC Vice Chair of ICPADS'2002; and as TPC Member of numerous IEEE and ACM conferences. He is the Vice Chair of IEEE TCPP and TCCC Executive Committees.received the Best Paper Awards in ACM MobiCom'99, ICOIN'01, ACM MSWIM'00, and ACM/IEEE PADS'97. He is also a recipient of the UTA's Outstanding Faculty Research Award in Computer Science in 2001 and 2003, and UTA's College of Engineering Excellence in Research Award in 2003.