Mobile Information Systems 6 (2010) 293–324
DOI 10.3233/MIS-2010-0105
IOS Press

Extending mobility to publish/subscribe systems using a pro-active caching approach

Abdulbaset Gaddah and Thomas Kunz[*]
Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada

Abstract. The publish/subscribe communication paradigm has many characteristics that lend themselves well to mobile wireless networks. Our research investigates the extension of current publish/subscribe systems to support subscriber mobility in such networks. We present a novel mobility management scheme based on a *pro-active* caching approach to overcome the challenges and the performance concerns of disconnected operations in publish/subscribe systems. We discuss the mechanism of our proposed scheme and present a comprehensive experimental evaluation of our approach and alternative state-of-the-art solutions based on *reactive* approaches and *durable subscriptions*. The obtained results illustrate significant performance benefits of our proposed scheme across a range of scenarios. We conclude our work by discussing a modeling approach that can be used to extrapolate the performance of our approach in a near-size environment (in terms of broker network and/or subscriber population) to our experimental testbed.

Keywords: Message-oriented middleware, publish/subscribe paradigm, mobility management, mobile computing, wireless networks

1. Introduction

A publish/subscribe (pub/sub) system is a push-based information dissemination model that inherently decouples communication between publishers and subscribers in *time*, *space*, and *flow* [2,13]. In such a system, *publishers* are the information producers that deliver information to a distributed set of brokers in the form of *messages* (or*events*), *subscribers* are the information consumers that subscribe to receive a selective set of messages within the system, and *brokers* are the routers that ensure the reliable and timely delivery of published messages to all interested subscribers. The pub/sub-based architecture is recently considered as a promising communication paradigm for future mobile information dissemination applications [22,27]. This is due to the advantages of this paradigm, including *decoupling*, *anonymous*, and *asynchronous* many-to-many information dissemination.

Most existing pub/sub systems [8,9,39,54] are designed for fixed wired networks, where both publisher and subscriber clients are usually stationary and have reliable low-latency high-bandwidth connections. Support and optimizations for client mobility are not built-in features of their formal semantics. Instead, it is left to the applications to adapt to the conditions of dynamic environments. This can significantly complicate the development of information dissemination applications. Recently, some literature [11, 42,46] has taken a first step towards supporting mobility in pub/sub systems. There is hence a pressing need for add-on protocols to extend these systems to operate in mobile wireless environments that

[*]Corresponding author: Thomas Kunz, Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, ON, Canada K1S 5B6. Tel.: +1 613 520 3573; Fax: +1 613 520 5727; E-mail: tkunz@sce.carleton.ca.

are characterized by frequent and unpredictable disconnections of participants due to wireless channel impairments or client mobility.

Although mobility management is widely studied by mobile computing researchers, the indirect communication paradigm of pub/sub systems introduces new challenges in designing handoff management solutions [11,41]. In a pub/sub system, published messages do not rely on an explicit destination address set by the publishers. Instead, they are routed to the end points (*subscribers*) based on their content and the subscriptions in the system. In other words, publishers do not know the explicit addresses of subscribers and therefore acknowledgement mechanism cannot be used to identify message loss. Subscribers cannot also depend on the sequence numbers of the received messages to detect message loss as they receive a selective set of the published messages. As a result, without any coordination between network brokers, subscribers may miss some or all the messages that were published during their movements from one broker to another. This can be a serious issue for some applications that do not tolerate message loss.

Since the migration of the subscribers is transparent to the system, the network brokers end up managing a large number of inactive subscriptions and tracking their corresponding messages. As perpetually caching messages for migrated subscribers imposes a substantial overhead on the brokers, the overall system performance may gradually degrade to the point of failure. Moreover, the subscribers may receive duplicated messages when they reconnect to the previously visited brokers. Such duplication may result in flooding the wireless channel and wasting a considerable amount of the bandwidth. Reported studies [14,15,32] discuss the above issues in details. Thus, the handoff management solutions for pub/sub systems should take into account these factors in addition to the conventional objectives such as low handoff latency and message overhead to guarantee reliable message delivery semantic and to hide the interruption of message dissemination.

In recent years, several mobility management solutions [7,11,14,29,36,52] have been proposed for pub/sub systems deployed on various wireless environments. One most commonly-used solution is based on a *reactive* scheme [7,11,53]. The reactive scheme works as follows. Once a mobile subscriber disconnects from source broker B_i, the broker starts to locally store published messages that match the subscriber's subscriptions. When the subscriber reconnects to target broker B_j, it first informs B_j that it was previously connected to B_i. Then B_j contacts B_i to fetch the subscriptions associated with the mobile subscriber. After B_j obtains all the subscriptions, it subscribes these subscriptions and informs B_i to remove them. Then B_j begins to store in a temporary queue all the new messages it receives for the moving subscriber. Meanwhile, B_i sends all the subscriber messages to B_j. After all the messages are forwarded, B_j simply replays the set of locally stored messages and received messages from B_i to the subscriber, potentially after removing duplicates from both set of messages. This scheme may result in a drastic increase in the network traffic load since the subscriptions and actual messages need to be transferred between the brokers [5]. It also imposes high handoff latency that may not be acceptable by applications requiring fast handoffs between brokers to maintain high communication quality.

Another recently-used solution, based on a *durable subscription-based* approach [14,20,31,32,36], is proposed to cope with the connection/disconnection operations. This approach is believed to be highly reliable and is typically used for applications that cannot tolerate message loss. In the absence of any mobility management mechanism, the durable subscription-based approach suffers from the issues described early when it is deployed on a mobile wireless domain in addition to the fact that frequent mobility of subscribers significantly degrades the system performance. In such a scheme, as broker B_i has no knowledge about the state of the mobile subscriber, which has already reconnected to broker B_j, it will keep buffering messages for that subscriber, causing B_i a significant performance overhead.

Also, the durable subscription-based scheme does not support a mechanism that removes the subscriber subscriptions from the previously visited brokers. In this case, each broker will end up managing a large number of inactive subscribers that may not reconnect again to that broker. Therefore, the durable subscription-based scheme gains by not propagating subscriptions and messages between the brokers, at the expense of perpetually caching messages for inactive subscribers. Results reported in [14,15,20, 32] show that the system's performance gets increasingly worse as the population of inactive subscribers increases in the system.

In this paper, we propose a novel and efficient mobility management scheme for current pub/sub systems to support subscriber mobility and to provide fast handoffs. The core idea of this scheme is to intelligently transfer and cache subscriber contexts (its actual subscriptions) one broker-hop ahead of its current broker in a *pro-active* manner (i.e., context transfer/caching occurs before the subscriber movement). Since it is difficult to predict the subscriber's movement, we need to identify the *set* of potential next brokers without examining the brokers' topology and manually creating the set. We exploit a data structure, called *neighbor graph*, which forms the basis for our proposed pro-active scheme as it dynamically captures the potential mobility graph of mobile subscribers. Each broker over time learns about its immediate neighbors; thus, only these neighbors will receive/cache the subscriber context prior to the occurrence of handoffs.

We have comprehensively evaluated the performance of our proposed pro-active scheme through testbed experiments, comparing it to alternative solutions: reactive and/or durable subscription-based. The obtained results show that our pro-active scheme achieves superior performance across a range of scenarios over the other solutions in terms of message loss, message duplication, and handoff latency. Using two different mobility models, we demonstrate that the pro-active approach can outperform the other solutions even when the neighbor graph is a relatively weak predictor of mobility (i.e., each broker has many neighbors). As the neighbor graph narrows the choice of potential next-hop brokers, the performance improvements become even more noticeable. We conclude our work by discussing a modeling approach that can be used to extrapolate the performance of our proposed mobility management scheme in a near-size environment (in terms of broker network and/or subscriber population) to our experimental testbed.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 describes the system model and assumptions. Section 4 presents the proposed pro-active mobility management scheme. Section 5 describes the experimental setup and discuses the evaluation results. Section 6 applies a modeling approach to extrapolate the performance of our proposed pro-active scheme. Section 7 concludes the paper.

2. Related work

In the recent years, several mobility management solutions have been proposed for well-known distributed pub/sub systems, like JEDI, SIENA, REBECA, and ELVIN. JEDI [11] is one of the first pub/sub systems to add support for subscriber mobility that is based on explicit *moveIn* and *moveOut* operations. The mobile subscribers explicitly invoke these operations during the handoff process, which can be problematic if a wireless link breaks down suddenly due to physical mobility or interference. Also, JEDI adapts a hierarchical topology of event brokers, which has a potential performance bottleneck at the root node of the hierarchical tree [5].

SIENA [7] is a scalable messaging system that has been extended to support subscriber mobility. The extension is typically presented in the reactive fashion and evaluated in wired and GPRS-based networks.

Although their reported results have demonstrated the applicability of their mobility extension, they are limited to narrow evaluations of a single mobile subscriber roaming across the network. This limits the value of their results since their proposed extension never needs to transfer a large volume of messages between brokers.

REBECA [53] incorporates a reactive-based solution to support physical mobility in an acyclic event topology. The mobility support scheme uses an intermediate node between the source and target broker, called *Junction*, for synchronizing the brokers. The source broker routes subscriber messages through the Junction to reach the target broker, and then the subscriber. The proposed extension relies on tracking the Junction broker that significantly increases the handoff delay, particularly in a large-scale network, and the overhead on the Junction [35]. It has not been justified why subscribers cannot maintain the information about the source broker, necessitating an intermediary to manage mobility.

ELVIN [46] is an event-based system that supports disconnected operation using a central caching proxy server but does not support mobility between proxies. In a wide-area system, mobility support between proxies is needed and may also be useful for load balancing purpose. The central proxy server tends to become a performance bottleneck and the system is not scalable.

Farooq et al. [14] presented their experience in evaluating the performance of a commercial JMS-based pub/sub system in wired and mobile cellular networks. The nature of their work differs from ours since it mainly focuses on studying the impact of certain mobility factors on the performance of reactive and durable subscription-based schemes without proposing a new mobility management scheme.

Chelliah et al. [10] proposed a distance-based cache relocation scheme to support continuity of service in a cellular network. To some extent, their approach is similar to the one proposed in [25]. The cache is relocated to the next base station once the mobile user reaches the relocated point in the cell. This mechanism relies mainly on predicting the path of the mobile user accurately. A combined approach of LA [45] and PM [28] techniques is used for path prediction. A distance-based relocation scheme is used to identify the time at which the relocation has to be performed. Distance between the mobile unit and base station is regularly monitored to make such a decision. The cost of prediction methods is high as they are repeatedly applied to every individual mobile user that enters to a cell. They also show low accuracy, which is not justified, even for a small number of visited cells. Moving close to the boundary of a cell initiates unnecessary cache relocation. Due to the fact that relocation takes place just after the handoff, the mobile user may experience service disruption after entering the new cell.

Katsaros et al. [2] introduced a prototype based on SCRIBE [42], an overly multicast routing system, to support mobility in pub/sub systems. In their prototype architecture, a mobile client is connected to an Overly Access Router (OAR) through the currently associated Access Point (AP). Since the mobile clients move from one AP to another, it is possible that they will be served by a different OAR. In such a case, the mobile client should inform the new OAR about the publications of its interest in order for the OAR to join the appropriate trees. The cost of finding and joining the appropriate multicast trees should be investigated as it has a direct impact on the handoff latency. Also, mobility prediction is absent in their proposed solution and thus proactive multicast group joins cannot be achieved to reduce the handoff latency. Caching mechanisms should be utilized to improve the efficiency of their approach.

Podnar et al. [41] discussed a persistent notification scheme to support subscriber mobility. In this scheme, each broker maintains a list of the IDs for the events routed to its neighbor brokers and the interested subscribers. It also stores the published events in a persistent buffer according to their lifetime. When a mobile subscriber reconnects to a new broker, the subscriber will provide the IDs of the latest received events to the new broker in order to avoid duplicated events. This scheme constantly burdens neighbor brokers with event transfer/caching that leads to a significant degradation in the system performance. This is due to the lack of coordination between subscriber mobility and the caching process. The scheme also adds extra load on the brokers as they are required to maintain a large volume of IDs and validate the lifetime of a tremendous number of events.

Burcea et al. [5] deployed a simple handoff management scheme that is based on the successful prediction of the subscriber destination. In the proposed scheme, the subscriber context is transferred to the destination brokers once the mobile subscriber disconnects from the network. The entire context will then be removed from the source broker. This limits the proposed scheme to only support mobility but not disconnect/reconnect operations that may occur frequently due to the loss of connectivity. Our proposed approach takes into consideration such behavior and can handle it well. The proposed scheme may also not be adequate for supporting fast handoffs particularly in large-scale networks as the context pre-fetching takes place only after the mobile subscriber disconnects from the network. In our scheme, the subscriber context is always transferred prior to the subscriber movement to support fast handoffs. The authors have not evaluated their approach in the presence of multiple brokers, which severely limits the applicability of their approach.

Wang et al. [49] provided a handoff management scheme, which is called *multi-hop handoff* (MHH), to offer reliable and ordered delivery of messages to mobile subscribers with minimized cost (in terms of message loss and duplication). In MHH, when a mobile subscriber disconnects from the system, the new incoming messages will be buffered at the subscriber's last visited broker. Once the subscriber reconnects to a new broker, the subscriber context (subscriptions and messages) is migrated in parallel. The context is typically moved hop-by-hop along the path from the last visited broker to the new broker in a reactive manner. In general, the proposed scheme will introduce high handoff latency as it may take a long time for the new broker to receive the subscriber context upon its reconnection, particularly when the network is congested or is large.

Hu et al. [21] addressed subscriber mobility in a distributed content-based pub/sub system. They mainly focus on the transactional semantics required by a mobile subscriber (i.e., a subscriber who wishes to disconnect from a source broker and reconnect to a new broker in the overlay as part of a transaction). They identified the transactional semantics for a mobile subscriber and outlined the transactional concerns at various layers, focusing on the subscriber movement and routing protocol layers. The proposed solution requires the system to reconfigure and update the routing tables of all the brokers on the path from the source to the destination broker. Such a behavior in the high mobility scenarios can be very costly from a performance perspective.

Tarkoma et al. [47] discussed a formal discrete model to study the safety and cost of handoff protocols for both publisher and subscriber mobility in content-based routing networks. Three new properties are defined to improve mobility support in the pub/sub topologies, namely *overlay-based routing*, *rendezvous points*, and *completeness checking*. Overlay-based routing prevents the content-based flooding problem. It abstracts the communication used by the pub/sub system from the underlying network-level routing and enables the system to cope with network-level routing errors and node failures. Rendezvous points simplify mobility by allowing better coordination of topology updates. Completeness checking ensures that subscriptions and advertisements are fully established (propagated) in the topology. The reported results are limited to a single performance metric (i.e., message cost), which is not sufficient to completely evaluate the impact of different protocols and topologies in managing mobile pub/sub clients.

3. System model

This section gives an overview of the system model and outlines some of the properties and assumptions of the used pub/sub system. In our system model, the broker network is modeled as an undirected *general* *peer-to-peer* graph $G = (V, E)$, where V is the vertex set of all brokers, $V = \{B_1, \cdots, B_k\}$, and E is the set of unordered distinct pairs of edges (or *links*), $e = \{B_i, B_j\}$ where $B_i \neq B_j$. Two brokers B_i and $B_j \in V$ may communicate directly only if they are linked by an edge e. We formally say that B_i and B_j have a neighboring relationship if $\{B_i, B_j\} \in E$. We thus define the set of all B_i neighbors in G as follows: *Neighbor* $(B_i) = \{B_j\colon B_j \in V, (B_i, B_j) \in E\}$. The set of brokers in G are assumed to be distributed in the same geographical neighborhood region and the mobile subscribers may reconnect to the physically closest brokers. This is an essential assumption in order to support emerging location dependent services (i.e., messages routed to the subscribers based on their current locations) and reduce the network overheads. It is assumed that G is a static routing topology [7] that routes publications to all interested subscribers.

The traffic on G travels via reliable and authenticated communication mechanism on each edge e and is classified into two different categories: *control* and *content* messages. The control traffic consists of the subscription and mobility update messages that are mainly exchanged by peer brokers whereas the content traffic consists of the actual data that are forwarded from publishers to interested subscribers. Each broker maintains two tables: a Subscription Table (*ST*) and a Neighborhood Table (*NT*). The ST consists of a set of $\{(E, \textit{Sub})\}$ pairs that record the subscriptions (*Sub*) received from or delivered to the immediate neighbor brokers. The edges of the neighbor brokers are recorded into the NT as a $\{(E,T)\}$ pairs that are used for the control traffic routing. T is a timestamp used to ensure the correctness and freshness of the NT as discussed later.

In our system model, a subscriber handoff is performed between two brokers in the network: B_i and $B_j \in V$ B_i is the source broker and B_j is the destination broker that is usually unknown to the mobile subscriber. After the subscriber disconnects from B_i, the handoff is negotiated and a new connection is established with B_j. Some systems limit subscriber mobility between only border brokers, the leaves of a routing tree. In contrast, we allow mobile subscribers to commute between any two brokers. The edge e between B_i and B_j is unique since G is a peer-to-peer graph. We refer to an edge e as *active* if a subscriber logically moves across that edge. It is necessary to keep track of *inactive* edges as they drastically impact the system's performance due to the overhead of propagating unnecessary subscriptions and messages in the system. Hence, we regularly update NT to capture the active edges and eliminate inactive ones. Broker B_j uses the propagated subscriptions, received from broker B_i, to store messages for offline subscribers in a dedicated buffer. All stored messages will then be routed directly to the mobile subscriber once it hands off to B_j. Subscription covering [26] is a key technique to quench the subscription propagation and hence reduces the propagation overhead. We give no consideration to such technique due to its high cost in an environment characterized by frequent movement [21]. In the described model, the mobile subscribers do not change their subscriptions during the course of disconnect/reconnect operations, but they are free to change them when connected to a broker.

The work described in this paper is based on the content-based subscription system that enables subscribers to express their interests with a finer level of granularity. It is also based on the use of the flooding strategy to steer message delivery to the distributed brokers. This strategy is strongly recommended by [3,39] in highly mobile environments since mapping content-based pub/sub systems on top of IP Multicast (*topic-based*) results in an explosion of multicast groups. Like most previously discussed work on supporting subscriber mobility in pub/sub systems, our work assumes that there are no failures at the pub/sub routing layer (broker nodes and their links) as the development of fault-tolerant pub/sub protocols [23] is out of the scope of this paper. Security has also not been addressed in conjunction with our proposed mobility management scheme. We assume that previously proposed techniques [44,48] can be also applied to our approach. In this work, we are only interested in managing subscriber mobility and not concerned about publisher mobility that has been addressed by others [37]. Unlike subscribers, there is no specific information that the publishers would miss during the period of their handoffs.

4. A pro-active mobility management scheme

Although the lower layers (such as *link-layer* and *IP-layer*) are conceptually the "*right layers*" to express the context of mobility support, the lower-layer mobility protocols have not been widely accepted and deployed for several reasons [4,47]. The application-layer mobility solutions on the other hand can easily remove the major drawbacks of the lower-layer mobility protocols and offer better mobility solutions for the next-generation heterogeneous networks. This motivates our choice to solve the mobility problem at the application-layer.

The core idea of our pro-active mobility management scheme is to intelligently transfer/cache a subscriber's context (subscriptions/messages) one broker-hop ahead of its current broker in a *pro-active* fashion (i.e., context transfer/caching occurs prior to the handoff of the mobile subscriber). A *neighbor graph*, which forms the basis of our proposed scheme, is used to dynamically capture the candidate set of brokers to which subscriber context should be pro-actively transferred to and cached at.

We model subscriptions to be in either *active* or *passive* mode. When a subscription is in active mode, it is used for filtering incoming messages and only those messages that match its filter are either routed to the subscriber or cached locally for future use. A passive subscription, on the other hand, is simply ignored in the filtering process. Initially, a subscriber submits an active subscription to a broker to receive the messages of its interest.

4.1. The pro-active context distribution algorithm

The pro-active context distribution algorithm is at the core of our pro-active mobility management scheme and described here at a high level as separate cases that correspond to various *connection, disconnection,* and *handoff* scenarios. Figure 1 depicts a simplified finite state machine (FSM) diagram that more formally describes the protocol. The following notation is used throughout the description:

S: a subscriber who is potentially mobile.
B_j: a broker who is initially serving S.
B_i: a neighbor broker of B_j.
Sub(*S*): S' subscriptions.
Msgs(*S*): S' messages.
Neighbor(B_j): set of neighbor brokers of B_j.
Context(*S*): $\{$*Sub*(*S*)$+$*Msgs*(*S*)$\}$.
Timeout(*S*): a chosen timeout for managing *Context*(*S*). When it expires, *Context*(*S*) is garbage collected.

The following pseudo-code summarizes the algorithm, as executed on each broker:

Case 1: When S initially connects to broker B_j, B_j sends a passive copy of *Sub*(*S*) to each broker B_i $\in$*Neighbor*(B_j), which locally stores *Sub*(*S*). In the meantime, broker B_j routes the published messages to S.

Case 2: When S disconnects from the network due to poor network connectivity or a handoff, broker B_j detects this (through receiving generic ping replies from S periodically) and consequently sends an activate request to each broker $B_i \in$ *Neighbor*(B_j). Following this request, broker B_j forwards *Msgs*(*S*) to its neighbors B_i until the activation request is acknowledged, to avoid message loss that may occur due to the activation latency. As *Sub*(*S*) is activated, *Neighbor*(B_j) will locally store all the incoming messages that match *Sub*(*S*). The ID of the latest message consumed by S (for each subscription) is

Algorithm 1: Pro-active Context Distribution – *executed on broker (B_j)*

```
Case 1: Initial connection
          IF subscriber $S$ connects to $B_j$ THEN
             FOR all $B_i \in$ Neighbor($B_j$) DO
               Forward Sub($S$) to $B_i$
             ENDFOR
          ENDIF
Case 2: Subscriber disconnects
          IF subscriber $S$ disconnects from $B_j$ THEN
             FOR all $B_i \in$ Neighbor($B_j$) DO
               Activate Sub($S$) stored at $B_i$
               Forward Msgs($S$) to $B_i$ stored during Sub($S$) activation
             ENDFOR
          ENDIF
Case 3: Subscriber reconnects
          IF subscriber $S$ reconnects to $B_j$ THEN
             FOR all $B_i \in$ Neighbor($B_j$) DO
               Deactivate Sub($S$) stored at $B_i$
             ENDFOR
          ENDIF
Case 4: Subscriber moves out to a peer broker
          IF subscriber $S$ hands off to $B_k$ from $B_j$ THEN
             FOR all $B_i \in$ Neighbor($B_j$) and $B_i \neq B_k$ DO
               IF $B_i \notin$ Neighbor($B_k$)THEN
                  Delete Context($S$) stored at $B_i$
               ELSE
                  Deactivate Sub($S$) stored at $B_i$
               ENDIF
             ENDFOR
          ENDIF
Case 5: Subscriber moves in from a peer broker
          IF subscriber $S$ hands of to $B_j$ from $B_k$ THEN
             IF Sub($S$) is not in $B_j$ buffer THEN
               Obtain Context($S$) stored at $B_i$
             ENDIF
             FOR all $B_i \in$ Neighbor($B_j$) DO
               IF $B_i \notin$ Neighbor($B_k$)THEN
                  Forward Sub($S$) to $B_i$
               ENDIF
             ENDFOR
          ENDIF
Case 6: Subscriber unsubscribes
          IF subscriber $S$ unsubscribes from $B_j$ THEN
             FOR all $B_i \in$ Neighbor($B_j$) DO
               Delete Context($S$) stored at $B_i$
             ENDFOR
          ENDIF
Case 7: Subscriber context obtained
          IF $B_j$ obtains Context($S$) from neighbors THEN
             Buffer Context($S$) at $B_j$
          ENDIF
Case 8: Subscriber times out
          IF $B_j$ triggers Timeout($S$) THEN
             FOR all $B_i \in$ Neighbor($B_j$) DO
               Delete Context($S$) stored at $B_i$
             ENDFOR
          ENDIF
```

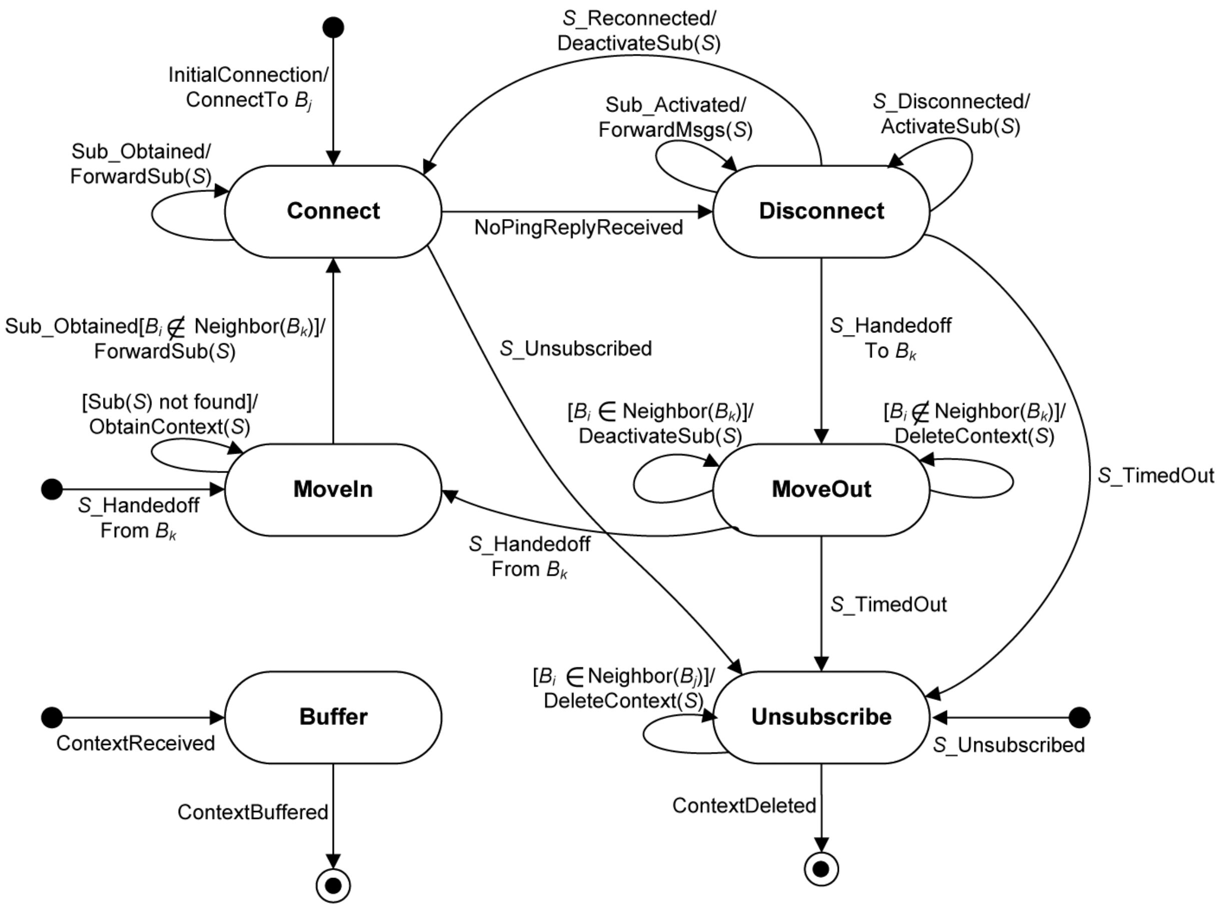


Fig. 1. FSM diagram for the pro-active context distribution algorithm.

enclosed with the activation request and thus only the messages with higher IDs are stored. Broker B_j similarly keeps buffering *Msgs*(*S*) as it may reconnect to B_j again.

Case 3: When S reconnects to the same broker B_j, rather than moving to a neighbor broker B_i, B_j sends a deactivate request to its neighbors B_i, informing them to deactivate *Sub*(*S*), terminate the caching process, and clean up their local buffers. In the meantime, broker B_j routes all buffered messages to S.

Case 4: When S moves from broker B_j to broker B_k, B_k informs B_j that is currently serving S. Accordingly, broker B_j requests its neighbors B_i either to delete *Context*(*S*) or deactivate *Sub*(*S*), excluding broker B_k. To reduce the overhead of context transfer,B_k and B_j exchange *Neighbor*(B_k) and *Neighbor*(B_j). Throughout these lists, broker B_j can decide which *Sub*(*S*) should be deleted and which *Sub*(*S*) should be deactivated for future use by broker B_k. Similarly, broker B_k can identify which *Sub*(*S*) should be routed to its neighbors B_i. Neighbor brokers typically exchange neighbor information whenever an edge is added to or deleted from their lists.

Case 5: When S moves from broker B_k to broker B_j, B_j first checks if *Context*(*S*) is available in its buffer. If it is not found in the buffer, B_j requests *Context*(*S*) from B_k. If *Context*(*S*) is found in the local buffer of B_j, similar actions to *case 1* will be performed.

Case 6: When S unsubscribes from broker B_j, B_j instructs its neighbors B_i to delete *Context*(*S*) from their buffers.

Case 7: When broker B_j receives *Context*(*S*) from its neighbors, it locally stores it in a persistent buffer.

Case 8: When S disconnects from broker B_j and *Timeout*(*S*) is reached, B_j informs its neighbors B_i to delete *Context*(*S*) from their buffers. This is a necessary task to manage mobile subscriber failures/crashes.

As this algorithm executes in parallel on all brokers, concurrency issues need to be dealt with, in particular potential race conditions due to network latency or/and delayed response of overloaded brokers. Race conditions appear when two concurrent operations, initiated by two different brokers, are intended to change the state of the same subscription(s) (e.g., activate, deactivate, or delete states). This could then lead to an inconsistent state among the same subscription(s) copies stored at neighbor brokers, and hence impact the performance of the pro-active approach. For example, when S enters the *MoveOut* state, broker B_j sends a deactivate request to the neighbor brokers B_i that are also neighbors of broker B_k. It may happen that S disconnects from B_k at the same time that B_j issued the deactivate request. In this case, B_k also needs to send an activate request to its neighbors B_i. Due to the previously mentioned delays, it is possible that the *Sub*(*S*) are activated to support the movement of S from B_k and then deactivated based on the request issued by B_j. To address this race condition, we have integrated the broker ID with the propagated subscription(s) to indicate which broker has the control to deactivate or delete the activated subscription(s). The integrated ID is frequently updated using the activate request received from the most recent broker that served the moving subscriber. Looking at the previous scenario, B_j would not be allowed to deactivate the *Sub*(*S*) since B_k gained control of the *Sub*(*S*) just after performing the activate request. This keeps the *Sub*(*S*) in a consistent/appropriate state and eliminates the overhead of the reactive method.

Similarly, race conditions may occur when S hands off to broker B_k. In this case, broker B_j needs to issue a delete request to remove the *Sub*(*S*) from its neighbor brokers B_i (that are currently not broker's B_k neighbors) once it is informed by B_k about the subscriber's handoff. It may happen that one or more of those brokers become broker's B_k neighbors just after notifying B_j. B_k does not need to propagate a copy of the *Sub*(*S*) to the new added neighbors as it has previous knowledge (through the exchanged neighbor information) that there are previously propagated copies of the same *Sub*(*S*) at those neighbors. Thus, if S disconnects from its current broker B_k and an activate request is sent by B_k to its neighbors, we may experience concurrent requests sent by B_k and B_j (activate and delete requests). The broker ID can again help to control such a situation. When the activate request is performed, broker B_k gains control over the *Sub*(*S*) and thus the delete request is ignored. Note that in case the activate request could not find the *Sub*(*S*), B_k will be notified and asked to deliver *Context*(*S*) prior to S arrival.

Finally, race conditions may occur in the *unsubscribe* state. If S hands off to broker B_k and just after the old broker B_j has been informed, S may decide to unsubscribe from the system. Thus, the neighbor brokers of B_j and B_k may receive concurrent requests (deactivate and delete). In this scenario, the delete request will fail if it gets executed before the deactivate request as broker B_j still has control over the active subscription(s). This leads to having a number of unclaimed subscriptions in the system. Note that the deactivate request disables the control attribute (broker ID) and therefore subscription(s) can be removed. To manage this race condition, the delete operation here has a special privilege to remove subscriptions without examining the control attribute of the subscriptions since the subscriber is leaving for good. A subscriber timing out is an alternative situation where a race condition may occur in the *unsubscribe* state. Consider the scenario when S disconnects from broker B_j and then hands off to broker B_k. If B_j for some reasons has not been informed about the subscriber movement, it will send a timeout request to the neighbors once the timeout interval is reached, deleting *Sub*(*S*). If S disconnects from B_k and at the same time B_j reaches its timeout interval, we again have conflicting concurrent requests (activate and delete). This race condition can be handled by using the control attribute (broker ID) as discussed earlier. Note that the delete request here does not have special privilege to ignore the control attribute as with the previous scenario.

4.2. Neighbor graph

The effectiveness of our proposed pro-active mobility management scheme largely relies on the successful approximation of the subscriber's movement. For a better chance of success, broker B_j can approximate a set of potential brokers that are most likely to be the next-hop destination of S. This approximation can be achieved, e.g., through observations of the mobility patterns of subscribers. We hence make full use of a data structure, called *neighbor graph*, which provides the abstractions to achieve this goal.

A neighbor graph is basically a geometrical representation of a network (a collection of vertices V connected in neighborhood fashion). The graph contains a set of edges E (or *mobility paths*) that directly connect every vertex v (broker) to each of its neighbors. As a result, the neighbors of a given broker B_i in the graph correspond to the set of potential next-hop brokers. The neighboring relationship among the brokers can be represented by *undirected* edges if it is reflective. In other words, if S travels from broker B_j to B_i or vice versa, we then connect B_j and B_i with a single undirected edge. As the mobile subscribers in our experimental testbed are allowed to travel in a bidirectional way, we choose to use an undirected graph to represent the mobility paths between the brokers.

One way of building the neighbor graph is to allow individual subscribers to capture their own mobility graphs and offer them to the brokers upon their connections. Building the graph in such a way has several drawbacks. Mobile subscribers presumably use portable devices with limited capability (in terms of CPU and memory) to interact with the distributed brokers in the backbone network. Hence, the task of capturing the mobility graph by subscribers adds additional load on these devices; potentially degrading their performance, especially in a large-scale network where the size of the global neighbor graph is large. Each mobile subscriber needs to repeatedly submit its global graph upon its connection to the target broker. This may result in consuming a considerable amount of bandwidth and potentially congesting the wireless channel, particularly with a large subscriber population. To approximate mobility in a neighborhood fashion, brokers need to acquire knowledge only about the local view of the complete graph (i.e., its subset of neighbor brokers). As a result, forwarding the global view to the target brokers is indeed inadequate as it may include a number of non-neighbor brokers, hence adding irrelevant load on the system. Every broker also needs to separately deal with (e.g., store, search, and update) the graphs of individual subscribers, which may complicate and increase the overhead of processing the pro-active approach. In addition, building subscriber-specific neighbor graphs will prevent subscribers to benefit from common movement patterns, which get reinforced as different subscribers migrate between specific brokers. Thus, we choose to allow individual brokers (typically running on machines with high capabilities) to automatically build the neighbor graph that captures the local view of their neighbors.

The neighbor graph can be constructed either in a static manner (i.e., manually created once and never changes over time) or in a dynamic manner (i.e., automatically generated and adaptively changes according to the mobility pattern). A static neighbor graph is problematic as it fails to adapt itself to the dynamic changes in the mobility pattern and/or broker topology. The neighbor graph also can be maintained either in a centralized manner (i.e., a single server stores the entire graph) or in a distributed manner (i.e., each broker stores a local view of the entire graph). A centralized neighbor graph raises

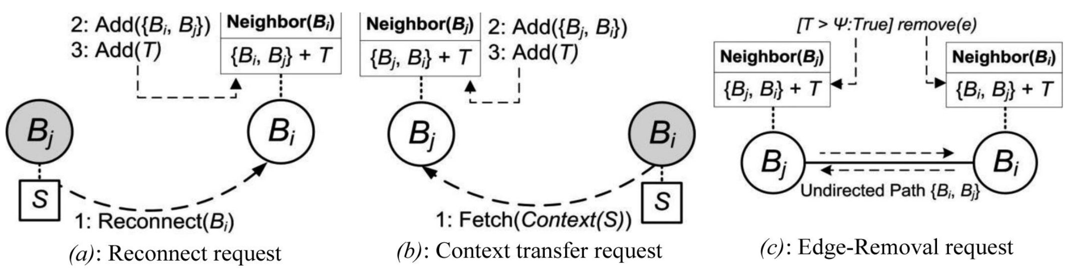


Fig. 2. The construction of the neighboring relation.

scalability and performance concerns as the network grows. Thus, we opted for a dynamic and distributed manner when generating the neighbor graph.

Several techniques [30,50] were proposed in the literature to build the neighbor graph, which are largely based on neighborhood search in the proximity space. In this paper, we adopt a different methodology for constructing the neighbor graph, which depends mainly on capturing the movement of mobile subscribers. Figure 2 illustrates how the adopted methodology works. Algorithm 2 summarizes the scenarios of the adopted methodology for constructing the neighboring relationship at each broker.

Algorithm 2: Construction of the Neighboring Relation – *executed at each broker*

```
Case a: Receive a reconnection request
      1: subscriber S reconnects to $B_i$ AND
         submits the address of $B_j$
      2: IF an edge e={$B_i$, $B_j$} is not in Neighbor($B_i$) THEN
            $B_i$ adds e in Neighbor($B_i$)
         ENDIF
      3: $B_i$ attaches a timestamp T to e OR
         updates an existing T
Case b: Receive a context transfer request
      1: $B_i$ requests $B_j$ to transfer the Context(S) AND
         encloses its address with the request
      2: IF an edge e={$B_j$, $B_i$} is not in Neighbor($B_j$) THEN
            $B_j$ adds e in Neighbor($B_j$)
         ENDIF
      3: $B_j$ attaches a timestamp T to e OR
         updates an existing T
Case c: Edge-removal request
      IF a timestamp T > a given time Ψ THEN
         remove an edge e from the neighborhood table
      ENDIF

      * Ψ ≥ the average frequency interval of edge-addition operations (cases
        a and b) at all broker nodes.
```

Two complementary methods can be used by each broker to effectively learn the edges in the neighbor graph. The *first* method is to attach the address of the old broker B_j with the reconnection request sent by the mobile subscriber S to the new broker B_i, hence building the neighboring relationship between the two brokers. The *second* method is to use the request for context transfer received from the new broker B_i to establish the relationship. This request is usually triggered whenever the *Context*(*S*) is not found at that broker. It is worth mentioning that some *outlier* edges (i.e., the ones that do not correctly model the neighboring relation) may be added to the neighborhood table. The table may also hold some *unused* edges that are created through rarely used paths. The impact of the outlier and unused edges on the performance of our pro-active approach can be significant due to the additional overhead required to cache *Context*(*S*) over time. It is thus essential to remove such edges from the graph table over time. To this end, we use a timestamp-based Least Recently Used (LRU) method to ensure the correctness and freshness of the graph. It should be apparent from that description that the autonomous creation of the graph makes it self-adaptive to dynamism in the neighboring relation (e.g., adding and deleting brokers, changing network topology, changing user behavior, etc.). Each broker independently builds and locally stores a subgraph of the complete graph of all brokers.

As the neighbor graph is initially an empty graph, the majority of handoffs, based on our creation algorithm, cause edge-addition during the early age of the graph, thereby reducing its benefit in our pro-active approach. Also, a mobile subscriber performing the first handoff along a path not in the graph may miss its messages, as the graph fails to provide information about the potential next brokers. To avoid this, the first mobile subscriber to cross over a path will receive its context in *reactive* manner. This will be gradually changed to the pro-active manner as the edges are added to the graph.

5. Performance evaluation

In this section, we describe our experimental environment and performance results for our pro-active scheme with various workload conditions. We compare our proposed scheme with two major existing schemes: *reactive* and *durable subscription-based*.

5.1. Experimental environment

The selected pub/sub system for our experimental study is based on a recent middleware technology called Java Message Service (JMS) [31]. JMS is a service-oriented API specification introduced by Sun to provide a standard platform for Java applications to create, send, and receive messages. Detailed descriptions of the JMS features can be found in [31,36]. We extended the selected JMS-based pub/sub system with the core functionalities of the pro-active and reactive schemes, the implementation details are more fully described in [17]. The durable subscription-based scheme is a built-in feature of all JMS-based pub/sub systems.

5.1.1. Testbed

Our testbed consisted of a dedicated network of *ten* Intel based Pentium 4 machines running RedHat 9, inter-connected by a 100 Mbps switch. Six machines were used for running six instances of the JMS broker with default configuration values. This work considers the distribution of these brokers in a simple in-building scenario as shown in Fig. 3 (left-side). The dotted lines represent a potential path of motion and the square boxes show the placement of broker nodes. Figure 3 (right-side) shows the complete experimental network. A router machine was used for running a wireless network emulator NistNet [38], with all the communication between the subscribers and the brokers tunneled through this router. All the configuration parameters for NistNet, like packet delay, packet loss, packet duplication, and network bandwidth, were set to the most commonly used values reported for IEEE 802.11 wireless LAN networks [19,51]. One machine was used for running a single message publisher $P1$ to inject messages in the broker network. The remaining two machines ($S1$ and $S2$) were used for running

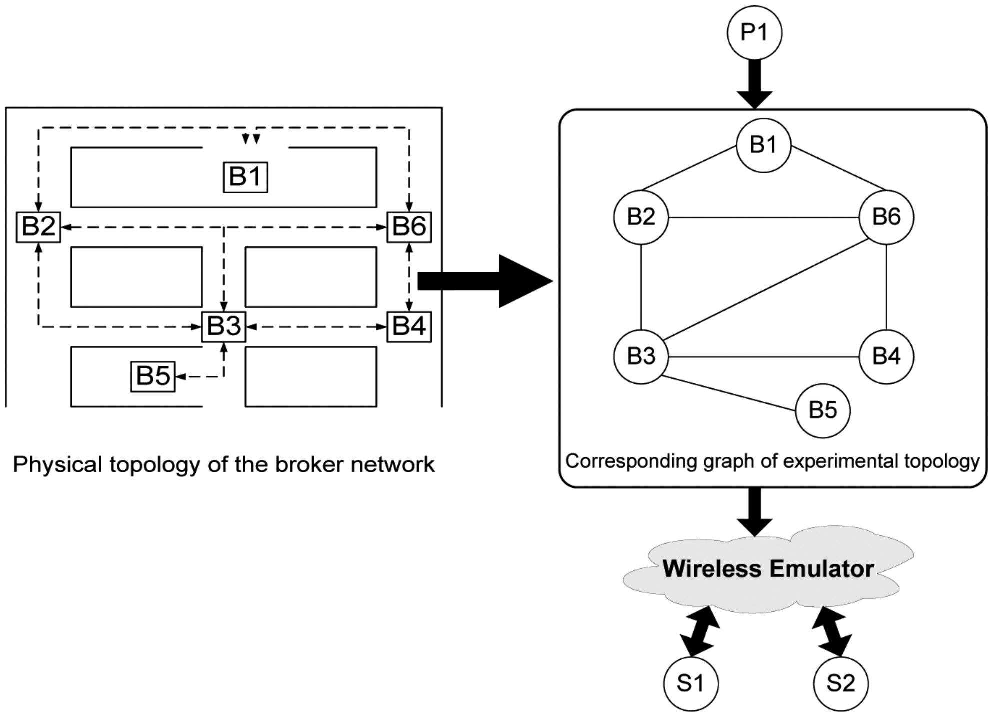

P1
B1
B2
B3
B4
B5
B6
Physical topology of the broker network
Corresponding graph of experimental topology
Wireless Emulator
S1
S2

Fig. 3. A general view of experimental setup constituting the network under consideration.

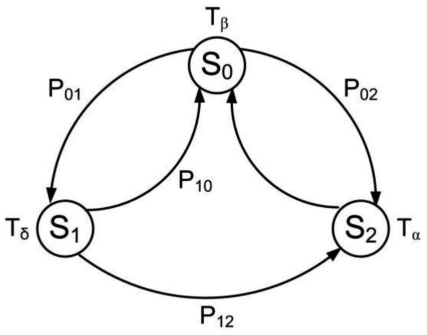


Fig. 4. Subscriber mobility model.

multiple subscribers. Subscribers that share the same machine run in separate threads and establish independent connections. Each subscriber subscribes to a single broker to receive approximately 20% of the published messages.

5.1.2. Mobility model

We developed a Java program to emulate subscriber mobility. Each subscriber goes through three mobility states: *connect* (S_0), *disconnect* (S_1), and *handoff* (S_2), as shown in Fig. 4. Initially, a subscriber enters to state S_0 to receive its messages from a uniformly chosen broker. The subscriber remains in S_0 for a randomly generated, exponentially distributed time with a mean of T_β seconds. With an equal probability $(P_{01} = P_{02})$, the subscriber either moves to state S_1 or S_2. S_1 reflects the case of signal

Table 1
Workload parameters

Workload parameters	Input values	Default values
Number of subscribers	10, 50, 100, 150, and 200	200
Sleep time	0, 0.5, 1, 1.5, and 2 seconds	0 seconds
Network bandwidth	1Mbps and 11Mbps	11 Mbps
Queue size	1, 2, 3, and 4 Mbytes	1 Mbytes
Mean disconnect interval (T_δ)	12 and 24 seconds	12 seconds
Mean connect interval (T_β)	60 and 150 seconds	60 seconds
Mean handoff interval (T_α)	1.5 and 3 seconds	3 seconds

breakdowns due to poor network connectivity. The subscriber remains in S_1 for a randomly generated, exponentially distributed time with a mean of T_δ seconds. With an equal probability $(P_{10} = P_{12})$, the subscriber either moves back to S_0 and reconnects to the same broker or goes to S_2. S_2 represents the case when a subscriber moves out of the covered area of its previous broker. After staying in S_2 for a randomly generated, exponentially distributed time with a mean of T_α seconds, the subscriber moves to S_0 and reconnects to a different broker.

We applied two mobility models, *random-based* and *neighboring-based* [1,12,18,24,33,43] to gauge how the three mobility management schemes react to different forms of subscriber mobility. In the random-based model, there are no dependencies or any other restriction modeled. The subscribers randomly move to any new target brokers deployed in an open geographical area. The target brokers are selected independently and uniformly, for every handoff, over the set of all six brokers. The subscribers pause at the selected brokers for randomly chosen time and then resume their movements. In the neighboring-based model, the area in which the subscribers are allowed to move is restricted due to obstructions. In other words, the subscribers can only move to some specific set of other locations from any given location, i.e., there are only a certain number of paths leaving each location. Thus, the subscribers in this model move to new neighbor brokers every handoff independently and uniformly over a specific set of neighbor brokers. Similarly, the subscribers pause at the selected neighbors for their randomly chosen time and then resume their movements.

5.1.3. Test conditions

Each experiment was run for a duration that was long enough to reach a steady state and repeated several times for verification purposes. Each broker machine was fully dedicated to run a single instance of the JMS broker. The CPU and memory usage of the broker machines were kept below 75% in any saturated modes (maximum publication rate and/or large subscriber population) to prevent performance bottlenecks. The publisher and subscriber machines must also not be bottlenecks, either, we ensured that their CPU and memory utilizations similarly stayed below 75%. We used the Linux tool "*sar*" to monitor the CPU and memory utilizations for each measurement run. Topic destinations and message stores are purged and reinitiated to start each test with a clean slate. Clock synchronization of the publisher and subscriber machines is required to calculate the end-to-end latency of message delivery and was done using NTP.

The reported results were captured from the measurement data obtained under different workloads, as summarized in Table 1. Unless otherwise stated, experiments were conducted using the default values listed in Table 1. The following metrics were used to evaluate the performance of the three mobility management schemes:

– *Subscriber throughput* (*Ts*): total number of messages received per second.
– *Broker throughput* (*Tb*): total number of messages routed per second to subscribers.

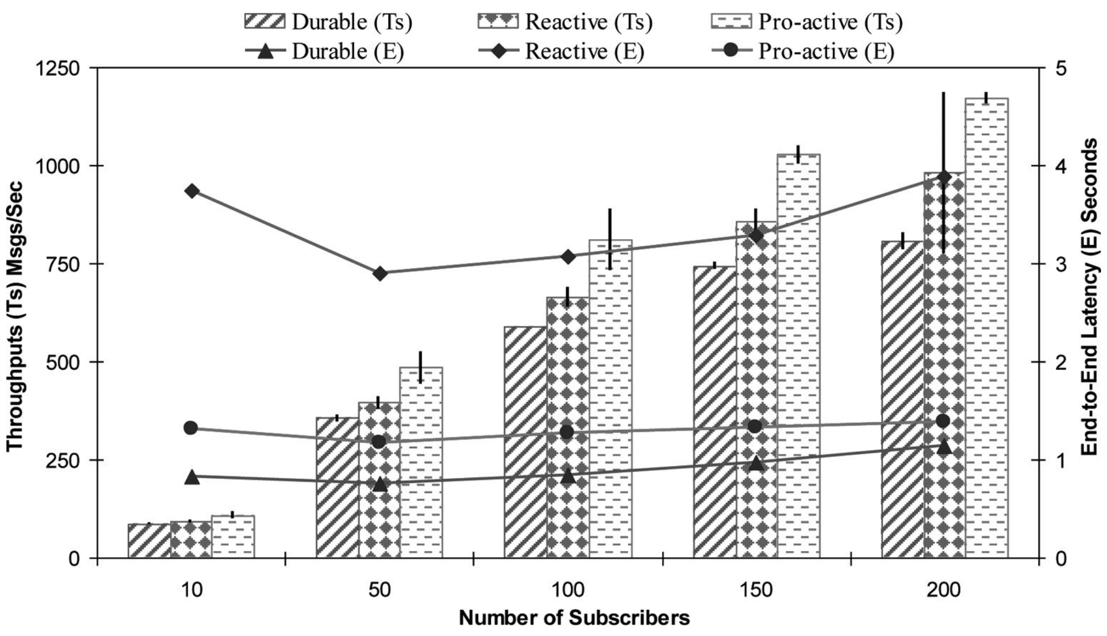


Fig. 5. Overhead of mobility support schemes.

– *Message loss* (L): percentage of missed messages by subscribers.
– *Message duplication* (D): percentage of duplicated messages received by subscribers.
– *End-to-end latency* (E): average time (in seconds) that it takes a message to travel from the publisher to the subscriber.
– *Handoff latency* (H): time (in seconds) between sending the reconnect request and receiving the first message of the subscriber at its new broker.
– *Message processing time* (M): average time (in milliseconds) that it takes a broker to filter messages and to route them to interested subscribers.

5.2. *Performance results for random mobility model*

We first present the performance results and comparisons of the pro-active, reactive, and durable subscription-based approaches in terms of overall subscriber throughput, end-to-end latency, handoff latency, message loss, and duplication. All the results presented are averages over 5 runs. For validity purposes, we plot the 95% confidence interval on top of each data point. The results were obtained using the random-based mobility model and default input values listed in Table 1. The random-based model presents the worse-case scenario for our pro-active approach: hard to predict mobility can result in the maximal protocol overhead. Due to space limitation only a representative set of the results are presented here. Interested readers are referred to [17] for more details.

5.2.1. *Overhead of mobility support*

Figure 5 shows how the three schemes compare in terms of end-to-end latency (E) and subscriber throughput (*Ts*) with the increase of subscriber population from 10 to 200. From the graph, we note that the *reactive* latency is by far the highest. A large portion of this latency is due to the overhead imposed by the state transfer semantic adopted by this scheme. During the handoffs, several messages may travel

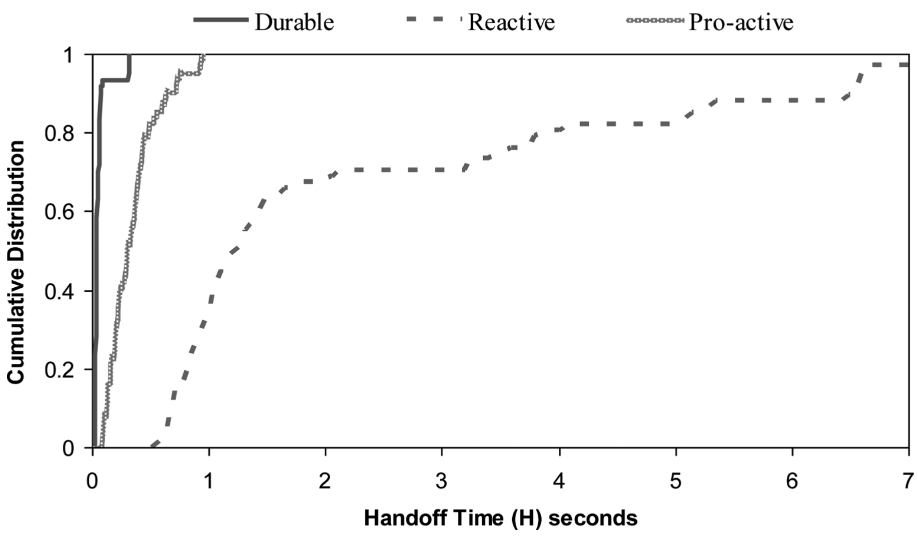


Fig. 6. Cumulative distribution of handoff times.

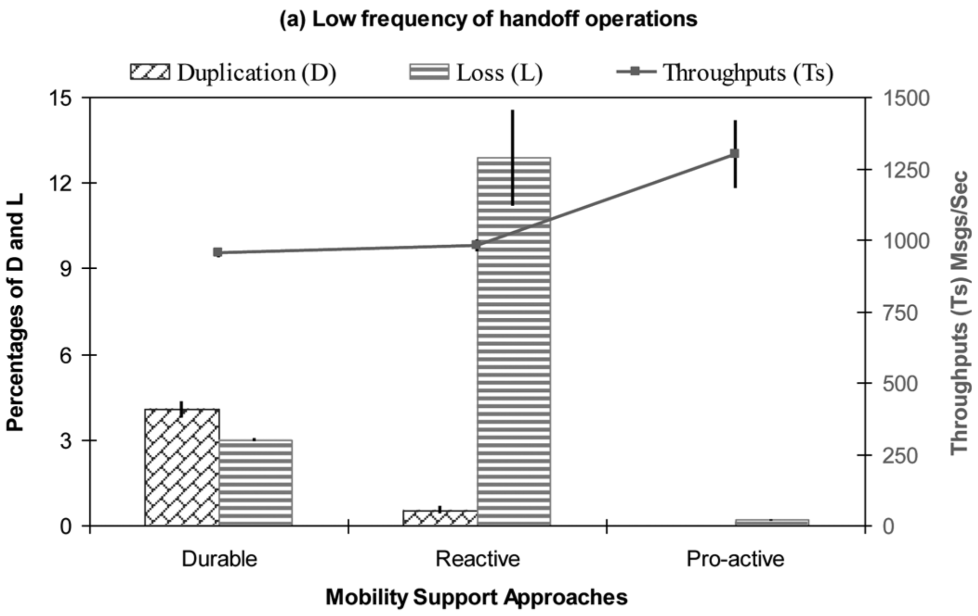
along multiple brokers as the state transfer protocol cannot catch up with the moving subscribers. We observed that the message overhead accounts for 37%–42% of the total consumed messages (i.e., the percentage of messages consumed via state transfer protocol) with 200 subscribers. The *pro-active* latency is much lower than the *reactive* as the *pro-active* scheme has almost no message overhead. In the *pro-active* scheme, messages are always available at the neighbor brokers and can be routed directly to the subscribers upon their reconnections. A small message overhead is imposed by this scheme to prevent message loss that may occur due to the latency of subscription activation request. The *durable-based* scheme shows the lowest latency as it has no state transfer overhead. However, it shows the lowest throughput results due to constantly caching messages at multiple brokers. The *pro-active* scheme also shows the highest throughput, as it caches messages on-demand and almost has no message overhead.

5.2.2. *Handoff latency*

Figure 6 shows the cumulative distribution of the handoff time. From the figure, we can note that the *reactive* scheme's latency is by far the highest among the three, with almost 60% of the handoffs taking more than 1.2 seconds. In contrast, almost 60% of the handoffs take less than 0.35 and 0.05 seconds with the *pro-active* and *durable-based* schemes, respectively. The observed results confirm that these two schemes can provide fast handoffs since the subscriber context is (almost) always ready at its new target broker prior to a subscriber's movement.

5.2.3. *Frequency of handoffs*

Figure 7 shows the performance results of the three schemes with low and high frequency of handoffs. We used a mean connect time interval (T_β) of 60 and 150 seconds in the low and high frequency of handoffs, respectively. From the figures, we observe that the overall throughputs of the *pro-active* and *durable-based* schemes slightly increase with the low handoff frequency. This is due to a reduced caching overhead in the two schemes. This can be clearly observed in the significant reduction of their message loss and duplication. Surprisingly, the overall throughput of the *reactive* scheme has not improved in the low frequency handoff scenario shown in Fig. 7(a), nor has its message loss ratio. In the high frequency handoff scenario, mobile subscribers connect with each broker for a short period. They may reconnect



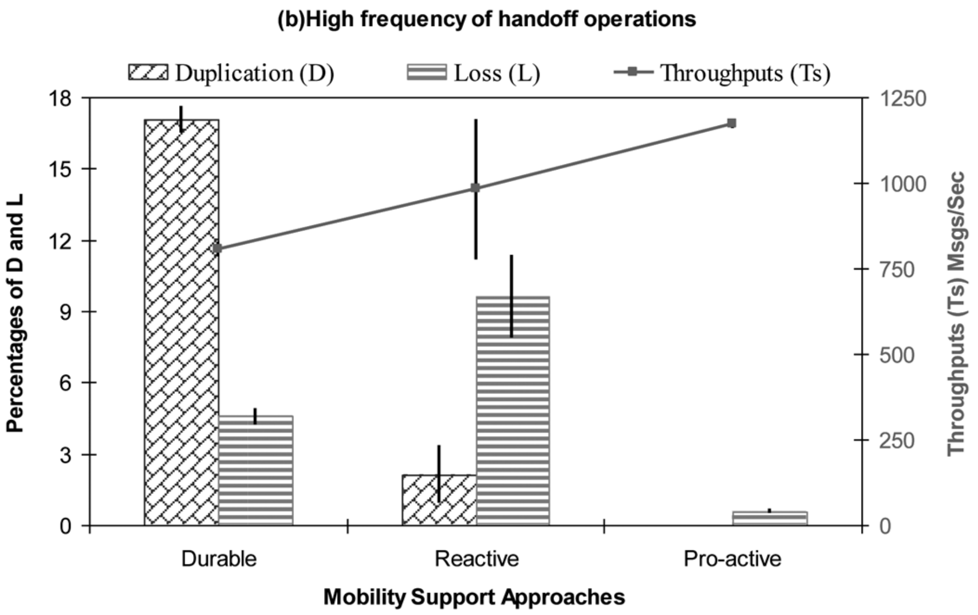


Fig. 7. Overall performance at a given handoff frequency.

back to their old brokers before the completion of their reactive state transfer. This leads to transferring a small number of messages between the old and new broker, but occurs quite often due to the high frequency of handoffs. In the scenarios with low handoff frequency, most mobile subscribers complete their state transfer before migrating to new locations. As a result, the full cost of the state transfer is incurred in such scenarios. For this reason, the throughput remains almost similar in the high and low frequency handoff scenarios. However, the low frequency handoff scenario shows higher message loss as the overhead of message transfer is high, with more messages discarded by the brokers.

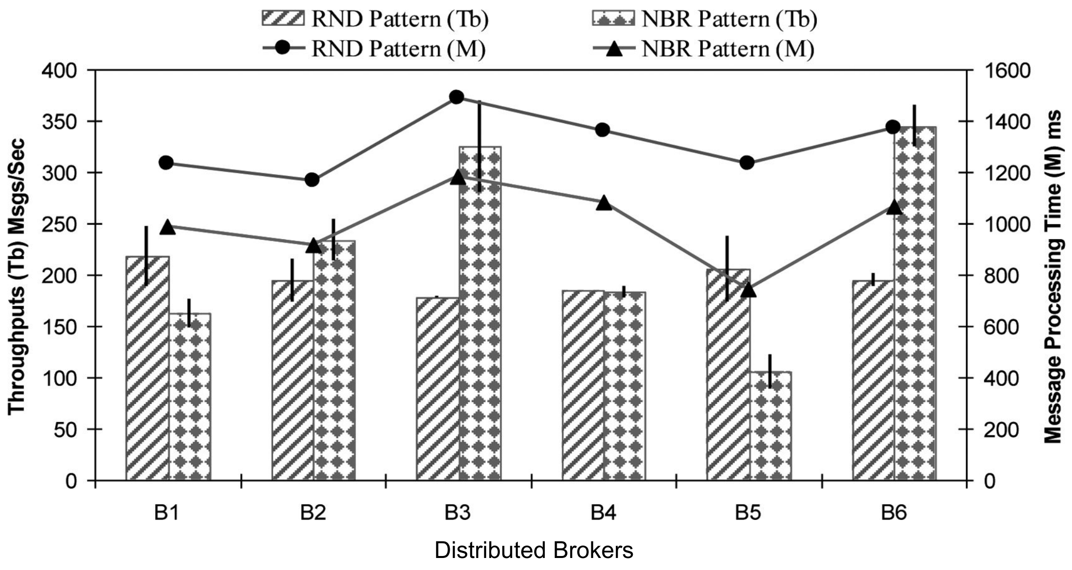


Fig. 8. Performance of pro-active approach in RND and NBR mobility patterns.

5.3. Comparing random and neighboring mobility patterns

Next we evaluate and compare the performance of the *pro-active* approach in the random (RND) and neighboring (NBR) mobility patterns in terms of message processing time (M) and individual broker throughput (*Tb*). Figure 8 shows the results of the selected metrics in the random (RND) and neighboring (NBR) mobility patterns: the left y-axis presents broker throughput (*Tb*) and the right y-axis shows message processing time (M). The x-axis category corresponds to the set of brokers presented in Fig. 3, describing the physical layout.

We observe that the *pro-active* approach shows lower message processing times under the NBR mobility pattern with all brokers. This is a good indicator of the overhead reduction due to limiting the mobility prediction to the (true) neighbor brokers. This results in reducing the number of immediate neighbors of each broker and therefore improves the performance of the *pro-active* approach. For example, broker $B5$ shows the lowest message processing time among all the brokers because it has only one neighbor broker $B3$. In the RND pattern, broker $B5$ shows higher processing time, as it has 5 neighbor brokers and needs to buffer messages for each disconnected subscriber at one of these neighbors. We can also observe that broker $B5$ experiences much lower throughput results in the NBR pattern. Since broker $B5$ has only one neighbor broker, the number of subscribers visiting broker $B5$ is much less than in the case of the random pattern. Therefore, broker $B5$ routes fewer messages to the overall subscribers. In contrast, brokers $B3$ and $B6$ have the largest number of neighbors (4 neighbors each) among all the brokers and show the highest throughput results. These brokers are visited by a large number of subscribers due to the NBR mobility pattern, thus many messages are routed through these brokers (traffic hotspots).

The remaining section presents and compares the performance of the three mobility management schemes using the RND and NBR mobility patterns and the default input values listed in Table 1. Three different performance metrics are used to evaluate the behavior of the three approaches: the overall subscriber throughput (*Ts*), message loss (L), and message duplication (D). We used different subscriber populations (10, 100, and 200) to investigate the overall performance of the three approaches under different workload conditions. Tables 2 through 4 summarize the respective results.

Table 2
Subscriber throughput (*Msgs/Sec*) in the RND and NBR mobility patterns

	Durable-based		Reactive		Pro-active	
Subscribers	RND	NBR	RND	NBR	RND	NBR
10	86.99	86.28	92.39	90.53	108.90	124.72
100	589.33	574.32	664.39	663.63	810.26	957.32
200	806.81	791.74	980.77	953.17	1172.77	1420.26

Table 3
Message loss (%) in the RND and NBR mobility patterns

	Durable-based		Reactive		Pro-active	
Subscribers	RND	NBR	RND	NBR	RND	NBR
10	4.78	3.13	3.65	5.14	2.25	1.26
100	5.14	4.06	8.92	7.03	0.86	0.72
200	4.56	3.51	9.62	9.44	0.58	0.51

As shown in Table 2, the *pro-active* approach shows superior throughput results under the RND and NBR mobility patterns, compared to the other approaches. As we expected, the *pro-active* approach under the NBR mobility pattern shows a noticeable improvement in the throughput results compared to the results achieved under the RND mobility pattern. This is because each broker in this pattern has fewer neighbors compared to the scenario in the RND pattern (brokers have probabilistically the same, maximum number of neighbors). As a result, the overhead (subscription propagation and message caching) of the *pro-active* approach significantly decreases and hence subscriber throughput improves. In contrast, the throughput results of the remaining approaches show a slight decrease with the various subscriber populations as shown in Table 2. We suspect that such behavior is due to the increased load on the central brokers (brokers that have a large number of immediate neighbors) that may become performance bottlenecks. In the *pro-active* approach, we found the overhead on the central brokers has not increased and is always less than the overhead under the RND pattern, as shown in Fig. 8.

Table 3 shows that the *pro-active*,*durable-based*, and *reactive* approaches demonstrate a slightly lower percentage of message loss under the NBR mobility pattern. This is due to the fact that some brokers' buffers (central brokers) are heavily utilized, but not others. Under the RND pattern, all the brokers' buffers experience almost similar (and high) buffer utilization. An interesting observation is that the percentage of message loss of the *pro-active* approach decreases gradually with the increase of subscriber population under both mobility patterns. With larger populations, the probability of having similar interest among the subscribers increases. This leads to a significant reduction in the caching overhead of the *pro-active* approach as one copy of each message can be stored for many subscribers. Therefore, message loss decreases with the increase of subscriber population. The *durable-based* approach shows an approximately similar percentage of message loss with the increase in the subscriber population. Although the *durable-based* approach benefits from the similarity of interest, its caching overhead is almost constant with different population sizes. This can be attributed to the continuous caching process adopted by this approach. In contrast, the *reactive* approach does not benefit from the similarity of interest as it does not reduce the overhead of state transfer. During the handoffs, state transfer has to be performed individually for every moving subscriber and hence its overhead increases proportionally with the subscriber population. Accordingly, message loss increases with the increase of the overhead imposed by subscriber population.

Table 4 shows that the message duplication slightly decreases under the NBR mobility pattern for the *durable-based* and *reactive* approaches. The pro-active approach shows zero message duplication under

Table 4
Message duplication (%) in the RND and NBR mobility patterns

	Durable-based		Reactive		Pro-active	
Subscribers	RND	NBR	RND	NBR	RND	NBR
10	11.35	12.10	0.80	1.52	0.0	0.0
100	12.01	11.68	1.70	0.85	0.0	0.0
200	17.05	16.97	2.13	1.90	0.0	0.0

both mobility patterns as it keeps track of the last consumed message by each subscriber and only buffers messages with higher ID than the last consumed message. Summarizing the results presented in these three tables, we can conclude that the NBR mobility model improves the performance of the *pro-active* approach in terms of overall throughput, message loss, and duplication. The two remaining approaches have not shown a significant difference in their performance results when using either the RND or the NBR mobility pattern.

6. Analytical approach for performance extrapolation

In this section, we present a modeling approach that can be used to extrapolate the performance of our proposed pro-active scheme in a near-size environment (in terms of broker network and/or subscriber population) to our experimental testbed. The general approach for performance extrapolation is as follows: most performance metrics are a function of the number of active subscribers at each broker. Thus, we first describe how to analytically derive the expected number of subscribers for a given broker topology, overall subscriber population, and mobility model. We then use a curve-fitting approach to relate the expected number of subscribers with a performance metric of interest: per-broker throughput. The approach can be generalized to other performance metrics. To validate our approach, the fitted curve derived from our experiments (using the random mobility model) is used to approximate the results in the neighboring mobility model.

Here we use continuous-time Markov chains (CTMC) to model subscriber mobility in the broker network topology presented in Fig. 3. This is due to the fact that we have a discrete state space of brokers (i.e., countable state space S $= \{1, \ldots,$H$\})$ and the *sojourn* times (holding time in one state before moving to another) are exponentially distributed. In the following two subsections, we describe in detail how CTMC is used for modeling the random and neighboring mobility of a mobile subscriber.

6.1. Modeling random mobility

In the random mobility model, a mobile subscriber has the freedom to randomly and uniformly move to one of N brokers available in the system. Thus, the state space S of this model can be defined by N *connect* states (brokers), a single *handoff*, and a single *disconnect* state. Hence, $S = \{1, \cdots, H\}$, where $H = N + 2$. The transition states of the random model can then be presented by an H-state Markov chain, as shown in Fig. 9.

In Fig. 9, states $\{1\}$ and $\{2\}$ represent the handoff and disconnect states, respectively. The rest of the states $\{3, \cdots, H\}$ correspond to the connect states. The arrows in the figure depict the subscriber mobility between different states while the parameters $\alpha, \delta,$ and β represent the transition (departure) rates from state$\{1\}$, $\{2\}$, and $\{3, \cdots, H\}$, respectively. Based on the state diagram in Fig. 9, the values of these

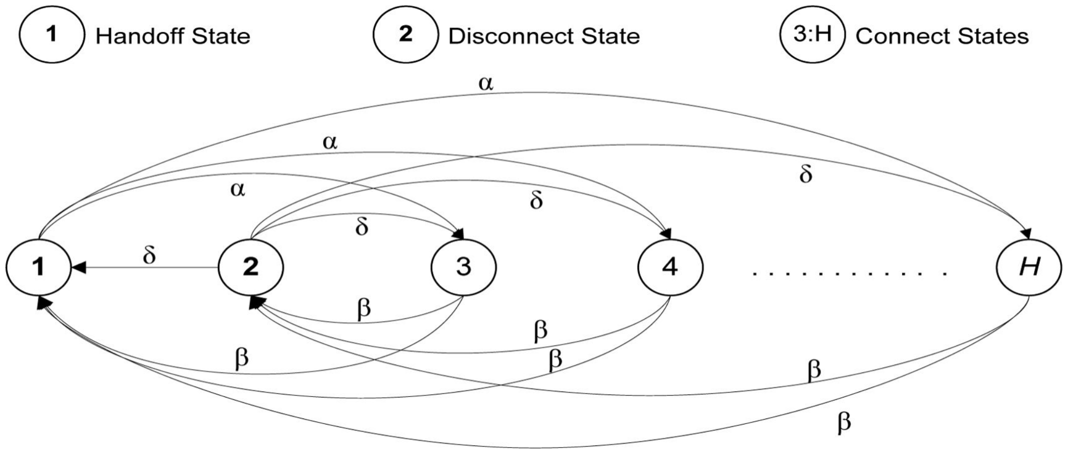


Fig. 9. State transition diagram for the random mobility model.

parameters can be determined as follows: Let T_α,T_δ, and T_β be the mean sojourn time at state$\{1\}$,$\{2\}$, and $\{3,\cdots,H\}$, respectively. Thus,

$$\alpha = \frac{1}{NT_\alpha} \tag{1}$$

$$\delta = \frac{1}{(N+1)T_\delta} \tag{2}$$

$$\beta = \frac{1}{2T_\beta} \tag{3}$$

The constant values (N, $N{+}1$, and 2) shown in the denominators of the previous three equations reflect the number of possible destination states from the current state.

The M-matrix (also known as infinitesimal generator matrix) of the space S according to the state transition diagram shown in Fig. 9 is described as follows:

$$M = \begin{bmatrix} -N\alpha & 0 & \alpha & \alpha & \alpha & \cdots & \alpha \\ \delta & -(N+1)\delta & \delta & \delta & \delta & \cdots & \delta \\ \beta & \beta & -2\beta & 0 & 0 & \cdots & 0 \\ \beta & \beta & 0 & -2\beta & 0 & \cdots & 0 \\ \beta & \beta & 0 & 0 & -2\beta & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta & \beta & 0 & 0 & 0 & \cdots & -2\beta \end{bmatrix}.$$

Let $\pi_i, i = 1,\cdots,H$, be the stationary state probability that the mobile subscriber is in state i. Accordingly, the H-element row vector $\pi = [\pi_1 \quad \pi_2 \quad \pi_3 \quad \cdots \quad \pi_H]$ represents the H stationary state probabilities that satisfy the matrix equation shown in Eq. (4) and their summation is equal to one, i.e., $\sum_{i=1}^{H} \pi_i = 1$.

$$\pi M = 0 \tag{4}$$

As the mobile subscriber moves to one of the connect states $i = 3, \cdots, H$ without any specific preferences and its mobility follows a symmetrical behavior in terms of arrival and departure, the stationary state probabilities $\pi_i, i = 3, \cdots, H$ are all equal. We denote the state probability of being in any connect state by P. Thus, we have $P = \pi_3 = \pi_4 = \cdots = \pi_H$, resulting in

$$\sum_{i=1}^{H} \pi_i = \pi_1 + \pi_2 + NP = 1 \tag{5}$$

By solving matrix Eq. (4), we obtain the state probabilities, π_1 and π_2, of the handoff and disconnect states, respectively.

$$\pi_1 = \frac{(N+2)\beta}{(N+1)\alpha} P \tag{6}$$

$$\pi_2 = \frac{N\beta}{(N+1)\delta} P \tag{7}$$

Substituting π_1 and π_2 into Eq. (4), we obtain the state probability, P, of being in any of the connect states i, $i = 3, \cdots, H$ as follows:

$$P = \left[\frac{(N+1)\alpha\delta}{N\beta\alpha + (N+2)\beta\delta + N(N+1)\alpha\delta} \right] \tag{8}$$

The expected number $E\left[x_i\right]$ of subscribers at any connect state i, $i = 3, \cdots, H$, follows the *binomial* distribution, since the presence of each subscriber at state i is a *Bernoulli* experiment with probability of success π_i and probability of failure $(1 - \pi_i)$. Let K be the total number of subscribers in the system. Thus, the expected number of subscribers at any connect state i, $i = 3, \cdots, H$, is given by *Pr*$\{x$ subscribers are at state $i\}$=$(\pi_i)^x \left(1 - \pi_i\right)^{K-x} \begin{pmatrix} K \\ x \end{pmatrix} = P^x \left(1 - P\right)^{K-x} \begin{pmatrix} K \\ x \end{pmatrix}$. This leads to

$$\overline{n}_i = E\left[x_i\right] = KP = K \frac{(N+1)\alpha\delta}{N\beta\alpha + (N+2)\beta\delta + N(N+1)\alpha\delta} \tag{9}$$

Similarly, we can obtain the expected number of subscribers at handoff state $\{1\}$, $\overline{n_1}$, and disconnect state $\{2\}$, $\overline{n_2}$. Hence, we have

$$\overline{n_1} = E\left[x_1\right] = K\pi_1 = K \left(\frac{(N+2)\beta\delta}{N\beta\alpha + (N+2)\beta\delta + N(N+1)\alpha\delta} \right) \tag{10}$$

$$\overline{n_2} = E\left[x_2\right] = K\pi_2 = K \left(\frac{N\beta\alpha}{N\beta\alpha + (N+2)\beta\delta + N(N+1)\alpha\delta} \right) \tag{11}$$

6.2. Modeling neighboring mobility

In the neighboring mobility model, a mobile subscriber, from its current state, can only move to one of its neighbor states, where the selected neighbor is chosen randomly and uniformly. The restriction on the possible brokers the subscriber can move to, from its current broker, is modeled by multiple disconnect and handoff states, a pair for each broker. Thus, the state space S of the neighboring model is defined by

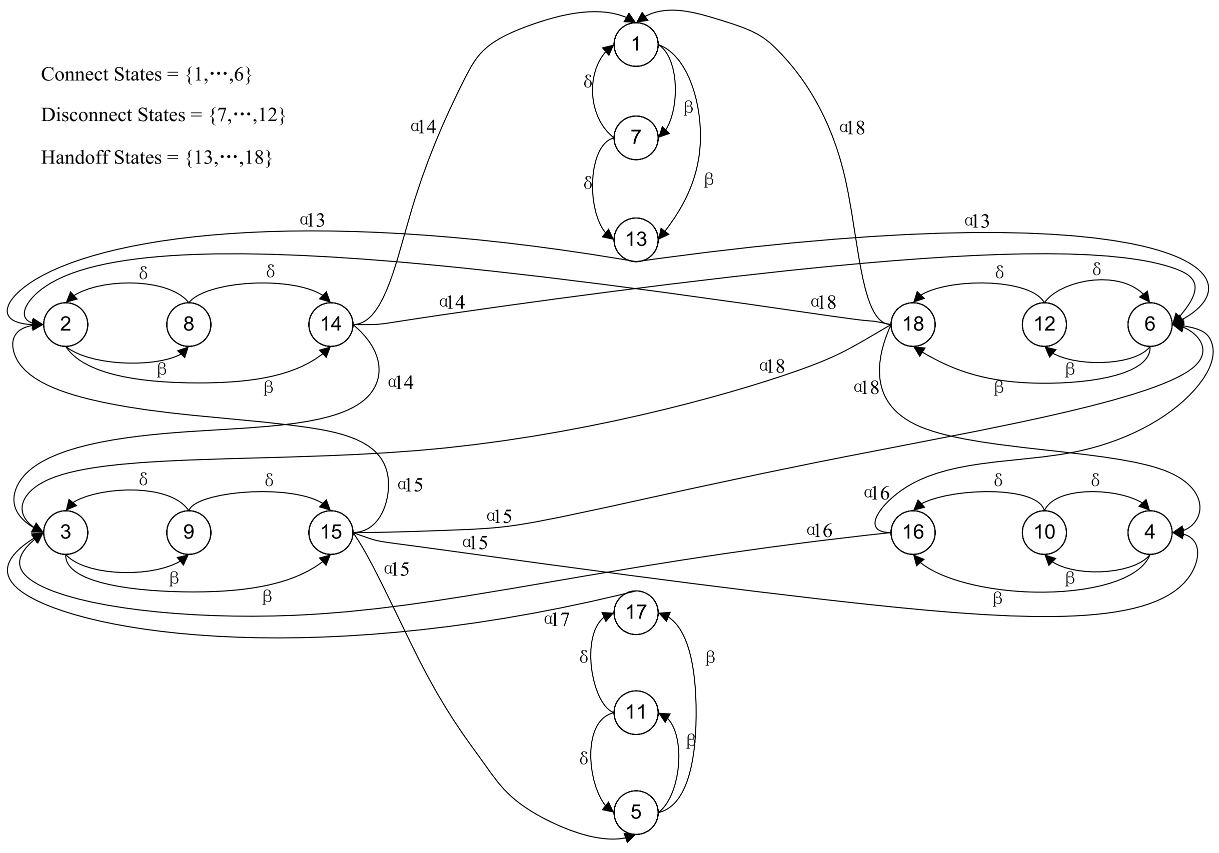


Fig. 10. State transition diagram for neighboring mobility model.

$S = \{1, \cdots, 18\}$, six connect states $\{1, \cdots, 6\}$, six disconnect states $\{7, \cdots, 12\}$, and six handoff states $\{13, \cdots, 18\}$. The transition states of the neighboring mobility model can accordingly be presented by the 18-state Markov chain, as shown in Fig. 10.

The arrows in the figure depict the subscriber mobility between different states while the parameters β, δ, and $\alpha_i, i = 13, \cdots, 18$ correspond to the transition (departure) rates from the connect, disconnect, and handoff states, respectively. Based on the transition state diagram shown in Fig. 10 and the mean sojourn time at each state, we can determine the values of the parameters (β, δ, and α_i) as follows:

$$\beta = \frac{1}{2T_\beta} \tag{12}$$

$$\delta = \frac{1}{2T_\delta} \tag{13}$$

$$\alpha_i = \frac{1}{E_i T_\alpha} \tag{14}$$

In the above three equations, the mean sojourn time at any connect, disconnect, and handoff states is denoted by T_β, T_δ, and T_α, respectively. The number of possible states from any connect or disconnect states is denoted by the constant 2 while the number of next possible neighbor states from any handoff state i, $i = 13, \cdots, 18$ is denoted by E_i.

We next determine the M-matrix of the state space S. As the cardinality of S is 18, the M-matrix has 18×18 entries that denote the transition rates based on the state transition diagram shown in Fig. 10. Thus, the M-matrix of the neighboring model is given by

$$M = \begin{bmatrix}
-2\beta & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 0 & 0 \\
0 & -2\beta & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 0 \\
0 & 0 & -2\beta & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 \\
0 & 0 & 0 & -2\beta & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 \\
0 & 0 & 0 & 0 & -2\beta & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 0 & 0 & \beta & 0 \\
0 & 0 & 0 & 0 & 0 & -2\beta & 0 & 0 & 0 & 0 & 0 & \beta & 0 & 0 & 0 & 0 & 0 & \beta \\
\delta & 0 & 0 & 0 & 0 & 0 & -2\delta & 0 & 0 & 0 & 0 & 0 & \delta & 0 & 0 & 0 & 0 & 0 \\
0 & \delta & 0 & 0 & 0 & 0 & 0 & -2\delta & 0 & 0 & 0 & 0 & 0 & \delta & 0 & 0 & 0 & 0 \\
0 & 0 & \delta & 0 & 0 & 0 & 0 & 0 & -2\delta & 0 & 0 & 0 & 0 & 0 & \delta & 0 & 0 & 0 \\
0 & 0 & 0 & \delta & 0 & 0 & 0 & 0 & 0 & -2\delta & 0 & 0 & 0 & 0 & 0 & \delta & 0 & 0 \\
0 & 0 & 0 & 0 & \delta & 0 & 0 & 0 & 0 & 0 & -2\delta & 0 & 0 & 0 & 0 & 0 & \delta & 0 \\
0 & 0 & 0 & 0 & 0 & \delta & 0 & 0 & 0 & 0 & 0 & -2\delta & 0 & 0 & 0 & 0 & 0 & \delta \\
0 & \alpha_{13} & 0 & 0 & 0 & \alpha_{13} & 0 & 0 & 0 & 0 & 0 & 0 & -2\alpha_{13} & 0 & 0 & 0 & 0 & 0 \\
\alpha_{14} & 0 & \alpha_{14} & 0 & 0 & \alpha_{14} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3\alpha_{14} & 0 & 0 & 0 & 0 \\
0 & \alpha_{15} & 0 & \alpha_{15} & \alpha_{15} & \alpha_{15} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4\alpha_{15} & 0 & 0 & 0 \\
0 & 0 & \alpha_{16} & 0 & 0 & \alpha_{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2\alpha_{16} & 0 & 0 \\
0 & 0 & \alpha_{17} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\alpha_{17} & 0 \\
\alpha_{18} & \alpha_{18} & \alpha_{18} & \alpha_{18} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4\alpha_{18}
\end{bmatrix}$$

Let $\pi_i, i = 1, \cdots, 18$, be the stationary state probability that the mobile subscriber is in state i. Thus, the 18-element row vector $\pi = [\pi_1 \quad \pi_2 \quad \pi_3 \quad \cdots \quad \pi_{18}]$ represents the 18 stationary state probabilities that satisfy the matrix equation in Eq. (4), and their summation is equal to one, i.e. $\sum\limits_{i=1}^{18} \pi_i = 1$. Note that $\pi_i, i = 1, \cdots, 6$, $\pi_i, i = 7, \cdots, 12$, and $\pi_i, i = 13, \cdots, 18$ are the state probabilities of the connect, disconnect, and handoff states, respectively. These state probabilities can be obtained by solving the matrix equation indicated in Eq. (4). Now we can readily calculate the expected number of subscribers, $E[x_i]$, at any state i, $i = 1, \cdots, 18$ in a similar way to that described in Section 6.1. Let K be the total number of subscribers in the system. Hence, the expected number of subscribers at any state i is given by

Pr$\{x$ subscribers are at state $i\}$ =$(\pi_i)^x (1 - \pi_i)^{K-x} \binom{K}{x}$. This leads to

$$\overline{n_i} = E[x_i] = K\pi_i \tag{15}$$

6.3. *Curve-fitting*

A curve-fitting approach is used to relate the average number of subscribers at a broker with a performance metric of interest (here per-broker throughput) using a single function generated from some observed data. Therefore, the generated function can be used to numerically extrapolate near-future outcomes. There are several curve fit forms we could chose from to build a function that gives the "best"

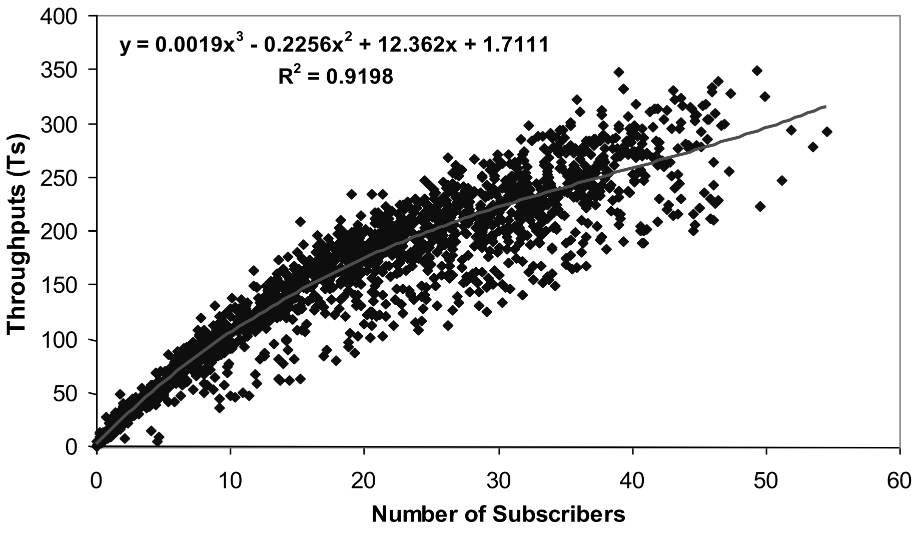


Fig. 11. Polynomial fit.

fit (i.e., the curve with minimum error between the generated curve and data points, usually referred to as *least-square error*). In this paper, we have chosen the *polynomial* curve-fitting approach since it shows the best fit, among the other generic forms we have tried, for our observed data. Using Excel, we fit a 3rd degree polynomial curve. Note that the observed data used for generating the curve is collected from the experiments of our general mobility model (random model). It is also collected from all the brokers used in our experimental setup. Figure 11 depicts the generated polynomial fitting-curve.

From this figure, we obtain the following polynomial equation that can be used to extrapolate the throughput of individual brokers as a function of the expected number of subscribers x.

$$y = (0.0019)x^3 - (0.2256)x^2 + (12.362)x + 1.7111 \tag{16}$$

The value of $R^2 = 0.9198$, depicted in the graph, is an indicator from 0 to 1 that reveals how closely the estimated values for the fitting-curve correspond to the observed data. A fitting-curve is more reliable when its R^2 (known as *R-squared* or the coefficient of determination) value is at or near 1.

6.4. Comparative study

Next we apply our analytical approach to derive the approximated per-broker throughput results in the *random* and *neighboring* mobility models, and compare them to our experimental results. We first determine the expected number $E(x)$ of subscribers at each broker in both models using CTMC. We then derive the approximated throughput results via Eq. (16) for each individual broker. For all the results reported next, we used the default values shown in Table 1 for T_β, T_δ, and T_α. These values correspond to the used values in our experimental setup. Unless otherwise stated, the total number of subscribers K in the system was set to 200 and the total number of brokers N was set to 6. Thus, the size of the state space S is $H = N + 2 = 8$ in the random model and 18 in the neighboring model.

Random Model Results: Using the default mean sojourn times indicated in Table 1, we can identify the departure rate from each state. From Eqs (1), (2) and (3), we get

$$\alpha = 1/18,\ \delta = 1/84,\ \text{and}\ \beta = 1/120$$

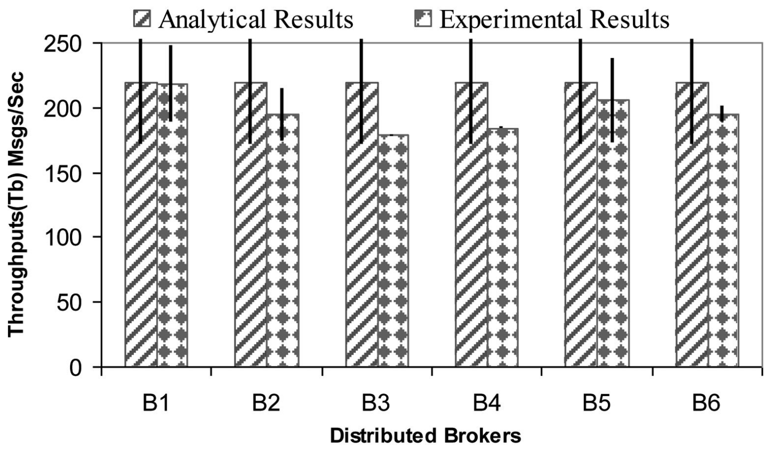


Fig. 12. Per-broker throughput results in the random mobility model.

We now describe the numerical solution for the set of Eqs (9), (10) and (11) using the obtained departure rates $(\alpha, \delta, \text{and} \beta)$. We first use Eq. (11) to obtain the state probability of being in connect state P, which is equal for all connect states as discussed earlier. Thus, we have $P = 0.147679$. Similarly, we can use Eqs (9) and (10) to obtain the state probability of being in the handoff and disconnect states, respectively. Hence, we have $\pi_1 = 0.025316$ and $\pi_2 = 0.088608$. We verify that the obtained state probabilities satisfy Eq. (5), i.e.,

$$\sum_{i=1}^{8} \pi_i = \pi_1 + \pi_2 + NP = 0.025316 + 0.088608 + (6)0.147679 = 1.$$

For a validity check of our analytical model, we have compared the analytical and experimental results in terms of the expected number of subscribers at each broker. Due to space limitations, these results are not shown here but were very close. Further details can be found in [17].

We next describe the numerical results obtained from the random model presented in Section 6.1. Based on Eq. (9), the expected number of subscribers/broker for a total subscriber population of 200 is obtained by $\bar{n}_i = 200 \times 0.147679 = 29.5358$. The expected throughput of each broker is computed by substituting $\bar{n}_i$ into Eq. (16). Thus, the per-broker throughput is given by y $= 218.983$ msgs/sec. We plot y (the expected throughput result) along with the experimental throughput results for the random model in Fig. 12. The lines across the data bars in the next figures represent the upper and lower bound of a 95% confidence interval for throughput results.

Comparing the 95% confidence interval for both sets of results, we note that the confidence intervals of the experimental results overlap all the corresponding intervals of the analytical results. This indicates that the differences between the analytical and experimental results are not statistically significant. From Fig. 12, we also note that the analytical results show relatively higher results compared to the experimental results. This can be attributed to the fact that each broker will have a large number of *proxy* subscribers that buffer messages on behalf of the moving subscribers. As a result, the throughput of individual brokers (especially ones executing on machines with lower hardware configurations such as $B3$ and $B4$) can be noticeably affected due to the overhead imposed by the proxy subscribers. Our analytical model can be refined in the future to take the number of proxy subscribers at each broker into consideration.

Table 5
State probabilities and E(x) of subscribers

State probabilities	Expected number of subscribers
$\pi_1 = 0.1099$	$\overline{n}_1 = E(x_1) = 200 \times 0.1099 = 21.98$
$\pi_2 = 0.1648$	$\overline{n}_2 = E(x_2) = 200 \times 0.1648 = 32.96$
$\pi_3 = 0.2198$	$\overline{n}_3 = E(x_3) = 200 \times 0.2198 = 43.96$
$\pi_4 = 0.1099$	$\overline{n}_4 = E(x_4) = 200 \times 0.1099 = 21.98$
$\pi_5 = 0.0549$	$\overline{n}_5 = E(x_5) = 200 \times 0.0549 = 10.98$
$\pi_6 = 0.2198$	$\overline{n}_6 = E(x_6) = 200 \times 0.2198 = 43.96$
$\pi_{DH} = 0.1209$	$\overline{n}_{DH} = E(x_{DH}) = 200 \times 0.1209 = 24.18$

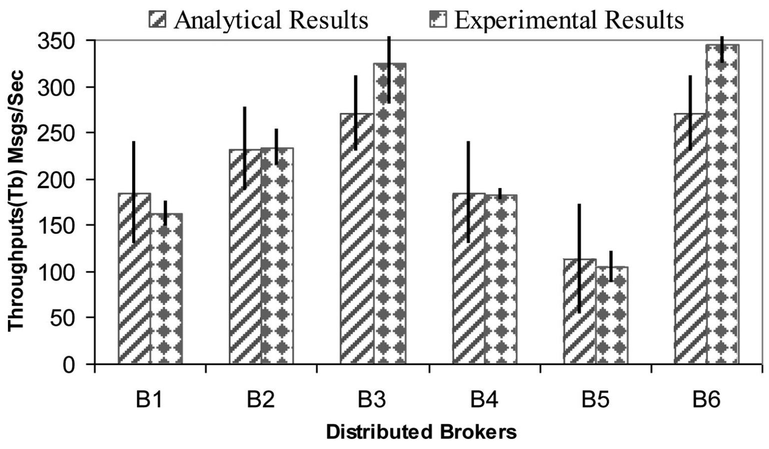


Fig. 13. Per-broker throughput results in the neighboring mobility model.

Neighboring Model Results: Similarly, we use the default mean sojourn times to determine the departure rate from each state. From Eqs (12), (13) and (14), we get

$$\beta = 1/120,\ \delta = 1/24,\ \text{ and } \alpha_{13\to18} = \{1/6, 1/9, 1/12, 1/6, 1/3, 1/12\}$$

We now describe the numerical solution for the matrix equation shown in Eq. (4) using the obtained departure rates (β, δ, and $\alpha_{13\to18}$). To solve Eq. (4), we plug in the rate values shown above in the $18 \times 18 M$-matrix presented earlier and then find the solution for Eq. (4). Solving Eq. (4), we obtain the state probability for the different states, and then the expected number of subscribers can be found by multiplying each state probability by the total number of subscribers K. Table 5 presents the state probabilities of the connect states $\pi_i, i = 1, \cdots, 6$ and the total state probability of disconnect and handoff states along with the expected number of subscribers for a total subscriber population of 200. Based on the obtained $\overline{n_i}$, the expected per-broker throughput results can be determined using Eq. (16). Figure 13 depicts a plot of the expected throughput results along with the corresponding experimental results obtained using the neighboring model.

From Fig. 13, we note that the analytical and experimental results are close for most brokers. We also observe that the analytical results show relatively lower throughput results with the central brokers, brokers with a large number of neighbors ($B3$ and $B6$), compared to the experimental results. Generally, the central brokers will be visited by a larger number of subscribers than other brokers, and from the curve shown in Fig. 11, we note that there is a greater variation across the experimental data points with the increase in the average number of subscribers. This may be the reason why brokers $B3$ and $B6$, which serve the highest average number of subscribers, experience the largest deviation between the analytical and experimental results. Another reason can be attributed to the fact that central brokers $B3$ and $B6$ have a larger number of proxy subscribers that buffer messages on behalf of the actual moving subscribers. Thus, the actual subscribers can consume more messages during their connect intervals, resulting in higher throughputs.

7. Conclusions and future work

In this paper, we proposed a novel and efficient *pro-active* mobility management scheme to extend current pub/sub systems to operate in mobile wireless settings. The proposed scheme pro-actively transfers subscriber context to the neighbor brokers prior to its movement to a new broker. The scheme depends largely on the use of a data structure, called *neighbor graph*, which dynamically captures the *set* of next potential brokers where the subscriber context should be transferred. The neighbor graph is automatically built and regularly updated to eliminate the outlier neighbors. We have comprehensively evaluated the performance of our proposed scheme through testbed experiments, comparing it to the durable subscription-based and reactive schemes. Our experimental results demonstrate that the pro-active approach is superior to the alternative solutions with respect to a number of performance metrics such as message loss, message duplication and system throughput. Though its handoff and end-to-end latency may be slightly higher than the durable subscription-base scheme, the pro-active scheme outperforms both alternatives in overall performance. While the performance advantages hold true for both mobility models, more predictable user movements (as in the neighboring mobility model) result in even better performance for our proposed scheme. We conclude our work by introducing a modeling approach to extrapolate the performance of our proposed pro-active approach in a near-size environment (in terms of broker network and/or subscriber population) to our experimental testbed.

A number of directions for future work exist. In our current model, we allow subscribers to choose which broker to attach to. From a system perspective, it may be preferable to let the pub/sub system make this decision, to balance the load between brokers. We are also currently not supporting publisher mobility. Finally, the efficiency of our proposed approach is very dependent on accurately predicting subscriber mobility. In an ideal situation, we would be able to narrow down the potential next-hop neighbors for each subscriber to a small set of target brokers. Newer proposals for handover prediction in cellular networks, such as [5,15,33] may therefore provide ways on further improving the performance of the proposed pro-active mobility management approach.

References

[1] N. Aschenbruck, E. Padilla and P. Martini, A Survey on Mobility Models for Performance Analysis in Tactical Mobile Networks, *Journal of Telecommunications and Information Technology* **2** (2008), 54–61.

[2] R. Baldoni, L. Querzoni, S. Tarkoma and A. Virgillito, Distributed Event Routing in Publish/Subscribe Systems, Middleware for Network Eccentric and Mobile Applications, ISBN 978-3-540-89706-4, Springer Berlin Heidelberg, 2009, 219–244.

[3] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. Strom and D. Sturman, An Efficient Multicast Protocol for Content-Based Publish/Subscribe Systems, In Proceedings of the 19th IEEE International Conference on Distributed Computing Systems (ICDCS'99), Washington, DC, 1999, 262–272.

[4] N. Banerjee, W. Wei and S. Das, Mobility Support in Wireless Internet, *Journal of IEEE Wireless Communications* **10**(5) (2003), 54–61.

[5] L. Barolli, A Speed-Aware Handover System for Wireless Cellular Networks Based on Fuzzy Logic, *Mobile Information Systems* **4**(1) (2008), 1–12.

[6] I. Burcea, H. Jacobsen, E. Lara, V. Muthusamy and M. Petrovic, Disconnected Operation in Publish/Subscribe Middleware", In Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM'04), Berkeley, California, 2004, 39–50.

[7] M. Caporuscio, A. Carzaniga and 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.

[8] A. Campailla, S. Chaki, E. Clarke, S. Jha and H. Veith, Efficient Filtering in Publish-Subscribe Systems Using Binary Decision Diagrams, In Proceedings of the 23rd International Conference on Software Engineering (ICSE'01), Toronto, Canada, 2001, 443–452.

[9] M. Castro, P. Druschel, A. Kermarrec and A. Rowstron, SCRIBE: A Large-Scale and Decentralized Application-Level Multicast Infrastructure, *IEEE Journal on Selected Areas in Communications,* (*JSAC'02*) **20**(8) (2002), 100–110.

[10] M. Chelliah, N. Govindaram and N. Gopalan, A Novel Distance Based Relocation Mechanism to Enhance the Performance of Proxy Cache in a Cellular Network, *The International Arab Journal of Information Technology* **6**(3) (2009), 258–263.

[11] G. Cugola, E. Nitto and 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.

[12] D. Engelhart, A. Sivasubramaniam, C. Barrett, M. Marathe, J. Smith and M. Morin, *A Spatial Analysis of Mobility Models: Application to Wireless Ad Hoc Network Simulation*, In Proceedings of the 37th Annual Simulation Symposium (ANSS'04), 2004, 35–43.

[13] P. Eugster, P. Felber, R. Guerraoui and A. Kermarrec, The Many Faces of Publish/Subscribe, *ACM Computing Surveys* **35**(2) (2003), 114–131.

[14] U. Farooq, S. Majumdar and E. Parsons, High Performance Middleware for Mobile Wireless Networks, *In Mobile Information Systems Journal* **3**(2) (2007), 107–132.

[15] P. Fülöp, S. Imre, S. Szabó and T. Szálka, Accurate Mobility Modeling and Location Prediction based on Pattern Analysis of Handover Series in Mobile Networks, *Mobile Information Systems* **5**(3) (2009), 255–289.

[16] A. Gaddah and T. Kunz, Performance of Pub/Sub Systems in Wired/Wireless Networks, *In Proceedings of the 64th IEEE Vehicular Technology Conference* (*VTC'06*), Montreal, Canada, 2006, 1–5.

[17] A. Gaddah, *A Pro-Active Mobility Management Scheme for Publish/Subscribe Middleware Systems*, Ph.D. Dissertation, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, 2008.

[18] S. Gowrishankar, T. Basavaraju and S. Sarkar, Effect of Random Mobility Models Pattern in Mobile Ad hoc Networks, *International Journal of Computer Science and Network Security* (*IJCSNS'07*) **7**(6) (2007), 160–164.

[19] N. Gupta and P.R. Kumar, A Performance Analysis of the IEEE 802.11 Wireless LAN Medium Access Control, *Communications in Information and Systems* **3**(4) (2003), 279–304.

[20] R. Henjes, D. Schlosser, M. Menth and V. Himmler, Throughput Performance of the ActiveMQ JMS Server, ITG/GI Symposium Communication in Distributed Systems (KiVS'07), Bern, Switzerland, 2007, 113–124.

[21] S. Hu, V. Muthusamy, G. Li and H. Jacobsen, Transactional Mobility in Distributed Content-Based Publish/Subscribe Systems, In Proceedings of the 29th IEEE International Conference on Distributed Computing Systems (ICDCS'09), Montreal, Canada, 2009, 101–110.

[22] Y. Huang and H. Garcia-Molina (2004), "Publish/Subscribe in a Mobile Environment", Wireless Networks Journal, Special Issue on Pervasive Computing and Communications, 10(6), pp. 643-652.

[23] H. Jafarpour, S. Mehrotra and N. Venkatasubramanian, A Fast and Robust Content-based Publish/Subscribe Architecture, In Proceedings of 7th IEEE International Symposium on Network Computing and Applications, (NCA'08), 2008, 52–59.

[24] A. Jardosh, E. BeldingRoyer, K. Almeroth and S. Suri, Towards Realistic Mobility Models for Mobile Ad hoc Networks, In Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom'03), San Diego, California, 2003, 217–229.

[25] K. Lai, Z. Tari and P. Bertok, Supporting User Mobility through Cache Relocation, *International Journal on Mobile Information Systems* (*MIS'05*) **1**(4) (2005), 275–307.

[26] G. Li, S. Hou and H. Jacobsen, A Unified Approach to Routing, Covering and Merging in Publish/Subscribe Systems Based on Modified Binary Decision Diagrams, In Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05), Washington, DC, 2005, 447–457.

[27] G. Li, V. Muthusamy and H. Jacobsen, Adaptive Content-Based Routing in General Overlay Topologies, In Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware (Middleware'08), Leuven, Belgium, 2008, 1–21.

[28] Y. Liu and G. Maguire, A Predictive Mobility Management Algorithm for Wireless Mobile Computing and Communications, In Proceedings of the 4th IEEE International Conference on Universal Personal Communications (ICUPC'95), Japan, 1995, 268–272.

[29] N. Liu, M. Liu, J. Zhu and H. Gong, A Community-Based Event Delivery Protocol in Publish/Subscribe Systems for Delay Tolerant Sensor Networks, *Journal of Sensors* **9**(10) (2009), 7580–7594.

[30] M. Maier, M. Hein and U. Luxburg, Optimal Construction of K-Nearest-Neighbor Graphs for Identifying Noisy Clusters, *Journal of Theoretical Computer Science* **410**(19) (2009), 1749–1764.

[31] Sun Microsystems, Java Message Service (JMS) API Specification, available on: http://java.sun.com/products/jms, 2009.

[32] M. Menth, R. Henjes, C. Zepfel and S. Gehrsitz, Throughput Performance of Popular JMS Servers, *SIGMETRICS Performance Evaluation Review* **34**(1) (2006), 367–368.

[33] G. Mino, L. Barolli, F. Xhafa, A. Durresi and A. Koyama, Implementation and Performance Evaluation of Two Fuzzy-Based Handover Systems for Wireless Cellular Networks, *Mobile Information Systems* **5**(4) (2009), 339–361.

[34] P. Mogre, M. Hollick, N. d'Heureuse, H. Heckel, T. Krop and R. Steinmetz, A Graph-Based Simple Mobility Model, In Proceedings of the 4th Workshop on Mobile Ad-Hoc Networks (WMAN '07), Bern, Switzerland, 2007, 421–432.

[35] G. Muhl, A. Ulbrich, K. Herrmann and T. Weis, Disseminating Information to Mobile Clients Using Publish-Subscribe, *Journal of IEEE Internet Computing* **8**(3) (2004), 46–53.

[36] M. Musolesi, C. Mascolo and S. Hailes, EMMA: Epidemic Messaging Middleware for Ad Hoc Networks, *Journal of Personal and Ubiquitous Computing* **10**(1) (2006), 28–36.

[37] V. Muthusamy, M. Petrovic, D. Gao and H. Jacobsen, Publisher Mobility in Distributed Publish/Subscribe Systems, In Proceedings of the 4th International Workshop on Distributed Event-Based Systems (ICDCSW'05), Washington, DC, IEEE Computer Society, 2005, 421–427.

[38] National Institute of Standards and Technology, NIST Network Emulation Tool, available on: http://snad.ncsl.nist.gov/itg/nistnet/index.html, 2009.

[39] L. Opyrchal, M. Astley, J. Auerbach, G. Banavar, R. Strom and D. Sturman, Exploiting IP Multicast in Content-Based Publish/Subscribe Systems, In Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware 2000), New York, NY, 2000, 185–207.

[40] P. Pietzuch and J. Bacon, Hermes: A Distributed Event-Based Middleware Architecture, In Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS'02) In Conjunction with the 22nd International Conference on Distributed Computing Systems (ICDCS'02), Vienna, Austria, 2002, 611–618.

[41] I. Podnar and I. Lovrek, Supporting Mobility with Persistent Notifications in Publish/Subscribe Systems, In Proceedings of the 3rd International Workshop on Distributed Event-Based Systems (DEBS'04), Edinburgh, Scotland, UK, 2004, 80–85.

[42] C. Rezende, A. Boukerche, B. Rocha and A. Loureiro, Understanding and Using Mobility on Publish/Subscribe Based Architectures for MANETs, In Proceedings of the 33rd IEEE Conference on Local Computer Networks (LCN'08), Montreal, Canada, 2008, 813–820.

[43] M. Saad and Z. Zukarnain, Performance Analysis of Random-Based Mobility Models in MANET Routing Protocol, *European Journal of Scientific Research* (*EJSR'09*) **32**(4) (2009), 444–454.

[44] M. Srivatsa and L. Liu, Securing Publish-Subscribe Overlay Services with EventGuard, In Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05), Alexandria, VA, USA, 2005, 289–298.

[45] H. Stathes and M. Lazaros, Using Proxy Cache Relocation to Accelerate Web Browsing in Wireless Mobile Communications, In Proceedings of the 10th ACM International Conference on World Wide Web, (WWW'01), Hong Kong, New York, NY, 2001, 26–35.

[46] P. Sutton, R. Arkins and B. Segall, Supporting Disconnectedness-Transparent Information Delivery for Mobile and Invisible Computing, In Proceedings of the 1st International Symposium on Cluster Computing and the Grid(CCGRID'01), Washington, DC, 2001, 277–285.

[47] S. Tarkoma and J. Kangasharju, On the Cost and Safety of Handoffs in Content-Based Routing Systems, *The International Journal of Computer and Telecommunications Networking* **51**(6) (2007), 1459–1482.

[48] C. Wang, A. Carzaniga, D. Evans and A. Wolf, Security Issues and Requirements for Internet-Scale Publish/Subscribe Systems, In Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02), Washington, DC, USA, 2002, 303–310.

[49] J. Wang, J. Cao, J. Li and J. Wu, MHH: A Novel Protocol for Mobility Management in Publish/Subscribe Systems, In Proceedings of the 2007 International Conference on Parallel Processing (ICPP'07), IEEE Computer Society, Washington, DC, 2007, 54–61.

[50] B. Wen, F. Liu, Z. Liu and F. Ma, Neighbor Relationship and Optimization in Mesh Overlay Multicast", In Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (ICHPCC'08), Washington, DC, 2008, 744–749.

[51] J. Yin, X. Wang and D. Agrawal, Modeling and Optimization for Wireless Local Area Network (WLAN), Computer Communications Journal, Special Issue on Performance Issues of Wireless LANs, PANs, and Ad Hoc Networks, **28**(10) (2005), 1204–1213.

[52] S. Yoo, J. Son and M. Kim, A Scalable Publish/Subscribe System for Large Mobile Ad Hoc Networks, *Journal of Systems and Software* **8**(2) (2009), 1152–1162.

[53] A. Zeidler and L. Fiege, Mobility Support with REBECA, In Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCSW'03), Providence, RI, 2003, 354–360.

[54] Y. Zhao and R. Strom, Exploiting Event Stream Interpretation in Publish-Subscribe Systems, In Proceedings of the 20th ACM Symposium on Principles of Distributed Computing (PODC'01), Newport, RI, 2001, 219–228.

Abdulbaset Gaddah recently completed his PhD in the Department of Systems and Computer Engineering at Carleton University. His research interests are in middleware for mobile networks, in particular publish/subscribe system. He also did research on proxy-based WWW access for resource-limited handheld devices.

Dr. Thomas Kunz is currently a Professor in Systems and Computer Engineering at Carleton University, Canada. He heads the Mobile Computing group, researching wireless network architectures (MANETs, wireless mesh networks, wireless sensor networks), network protocols (routing, Mobile IP, QoS support), and middleware layers for innovative wireless applications. Professor Kunz authored or co-authored over 150 technical papers, received a number of awards, and has served on over 40 TPCs of international conferences and workshops in the mobile and wireless domain.