J. Parallel Distrib. Comput. 69 (2009) 559–572

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

A proxy-based integrated cache consistency and mobility management scheme for client–server applications in Mobile IP systems

Weiping He *, Ing-Ray Chen

Department of Computer Science, Virginia Tech, Northern Virginia Center, 7054 Haycock Road, Falls Church, VA 22043, United States

ARTICLE INFO

Article history:
Received 1 June 2008
Received in revised form
4 November 2008
Accepted 20 February 2009
Available online 6 March 2009

Keywords:
Mobile IPv6
Cache consistency
Proxy
Regional registration
Integrated mobility and service management
Performance analysis
Petri net

ABSTRACT

In this paper, we investigate a proxy-based integrated cache consistency and mobility management scheme for supporting client–server applications in Mobile IP systems with the objective to minimize the overall network traffic generated. Our cache consistency management scheme is based on a stateful strategy by which cache invalidation messages are asynchronously sent by the server to a mobile host (MH) whenever data objects cached at the MH have been updated. We use a per-user proxy to buffer invalidation messages to allow the MH to disconnect arbitrarily and to reduce the number of uplink requests when the MH is reconnected. Moreover, the user proxy takes the responsibility of mobility management to further reduce the network traffic. We investigate a design by which the MH's proxy serves as a gateway foreign agent (GFA) as in the MIP Regional Registration protocol to keep track of the address of the MH in a region, with the proxy migrating with the MH when the MH crosses a regional area. We identify the optimal regional area size under which the overall network traffic cost, due to cache consistency management, mobility management, and query requests/replies, is minimized. The integrated cache consistency and mobility management scheme is demonstrated to outperform MIPv6, no-proxy and/or no-cache schemes, as well as a decoupled scheme that optimally but separately manages mobility and service activities in Mobile IPv6 environments.

1. Introduction

With the advances of IP-based wireless networks, and the growth in the number of mobile hosts (MHs) carrying wireless devices, it is widely speculated that Mobile IPv6 (MIPv6) [14] will become prevalent in next generation all-IP networks to allow users to maintain service continuity while on the go [19]. A major challenge is to maintain service continuity even if an MH disconnects and then reconnects at will. An MH may disconnect voluntarily simply to reduce the connection cost and/or to avoid power consumption. An MH may also disconnect involuntarily due to handoff or wireless link problems [20].

Two major sources of traffic in MIPv6 systems are mobility management [1,17] and service management [8]. Traditionally, mobility management has been considered separately from service management, as mobility management mainly deals with mobility handoff, location update and location search, while service management mainly deals with data delivery, and application servers can always query the underlying mobility management system to know the location of the MH. Over the years, many micro-mobility protocols extending MIP have been proposed with the goal to minimize mobility management overheads, including MIP Regional Registration [10,18] HMIPv6 [23], Cellular IP [25], IDMP [16], and HAWAII [21].

In this paper we consider integrated mobility and cache management in the context of MIPv6 environments, considering *cache management* as a form of service management. In particular, we propose a Proxy-based Integrated Cache and Mobility Management (PICMM) scheme in MIPv6 networks to support mobile client–server database applications in which the MH queries the server for dynamic data. For example, an MH may query dynamic data such as stock prices, dynamic web pages, weather reports, or traffic information. To avoid sending a query to the server and receiving a reply through the expensive and often unreliable wireless communication network, an MH can cache data objects on the local storage and then answer queries for data that are up-to-date. Caching reduces the server access cost and improves the user-perceived response time [12]. Cache management for such mobile client–server database applications essentially is service management since it deals with efficient data delivery issues.

To process user queries correctly based on caching, an MH must ensure that its cached data are up-to-date. Our integrated cache consistency and mobility management scheme proposed in this paper is based on the stateful strategy [24,27] by which a cache invalidation message is asynchronously sent by the server to the MH, whenever there is an update to a data object cached at the

* Corresponding author.
E-mail addresses: weiping@vt.edu (W. He), irchen@vt.edu (I.-R. Chen).

0743-7315/$ – see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2009.02.003

MH. The MH uses invalidation reports received to determine the validity of its cache content before answering a query. If a query asks for a data object that has been invalidated, then a request is sent uplink to the sever to ask for a fresh copy of the data object accessed by the query before the query can be answered. Moreover, to support service continuity in cases the MH is disconnected and then reconnected again, we use a per-user proxy to buffer invalidation messages to allow the MH to disconnect arbitrarily and to reduce the number of uplink requests to check the cache status when the MH is reconnected. The generated network traffic cost associated with cache consistency management thus includes the cost of receiving and buffering invalidation messages at the proxy and the cost of forwarding them from the proxy to the MH, as well as the cost of sending requests to the server and receiving responses in case cached data objects are not up-to-date to answer queries. This cost is considered as part of the "service" management cost which we like to minimize. Here we note that a large body of research in the literature on cache invalidation for mobile environments is based on the *stateless* strategy [31,13,5,30] by which the server has no knowledge of cache contents of MHs and will broadcast information on data objects that have been updated either periodically or asynchronously. However, the stateless strategy is not suitable for deployment over large wireless networks where users can roam from one network to another.

Our approach utilizes a per-user proxy to keep invalidation reports of the MH's cache. Whenever the proxy moves it informs the application servers of its whereabout, so invalidation reports can be sent directly from the application servers to the proxy. To minimize both the "service" and "mobility" network traffic costs, the proxy furthermore takes the responsibility of mobility management for integrated mobility and cache management. We investigate a design by which the MH's proxy serves as the MH's gateway foreign agent (GFA) as in the MIP Regional Registration micro-mobility protocol [10,18] to keep track of the address of the MH in a region. The proxy migrates with the MH when the MH crosses a regional area. We aim to identify the *optimal* regional area size under which the overall network traffic generated due to cache consistency management, mobility management, and query requests/replies is minimized. The idea can be extended to other micro-mobility management protocols such as HMIPv6 [23], Cellular IP [25], IDMP [16], and HAWAII [21], although in this paper we only consider the Regional Registration protocol.

The basic idea of our approach is that we use a client-side proxy to support caching and mobility management in Mobile IPv6. The proxy has three functions: (1) working as a GFA as in regional registration to keep track of the MH's location; (2) acting as a service proxy for services engaged by the MH; (3) allocating an extended cache space to store service context information including cache invalidation reports with timestamps for each MH. The proxy will receive invalidation reports from the server on behalf of the MH. If the MH is connected, the proxy will forward the invalidation report to the MH and then discard the invalidation report. If the MH is disconnected, the proxy will store invalidation reports in the proxy's extended cache. Once the MH is reconnected, the MH will get the latest invalidation reports from the proxy. The benefit of the proxy-based scheme lies in the fact that the proxy as an integral part of mobility management knows the MH's location information at all times and thus can efficiently manage the MH's cached data and perform data delivery on behalf of server applications.

The client-side proxy is created when the MH starts in MIPv6, acting as a GFA for the MH. We note that a cross-layer design such as MobileMAN [3] could be used for the implementation of the client-side proxy to run on the access router (AR). The cross-layer design has two components. The first component is at the network layer dealing with mobility management. The second component is at the application layer dealing with service management such as storing invalidation reports or service context information in its extended cache. We also note that Proxy Mobile IPv6 (PMIPv6) [15,9], a recent mobility management protocol for all-IP mobile networks, also proposes to run proxies on ARs for network-based mobility management. However only mobility management is addressed. The security issue of deploying proxies to run on ARs can be handled in a similar way as in PMIPv6. Finally, modern ARs are very powerful devices with ample memory space to run our proxies with extended cache functions. As the proxies of MHs (and hence their GFAs) will move from time to time as they cross regional areas, the load on ARs will be distributed according to the MHs' mobility patterns. In the case of random mobility, the load would spread out to ARs evenly.

The contributions of the paper are (1) the notion of integrated mobility and cache management to minimize the overall network traffic cost for supporting mobile client–server query applications in future MIPv6 systems; (2) identifying the optimal proxy setting including the region area size that will minimize the overall network traffic generated due to mobility and cache consistency management; and (3) demonstrating the benefit of integrated mobility and cache consistency management in MIPv6 compared with basic MIPv6, no-proxy and/or no-caching schemes, as well as a decoupled scheme under which mobility management and cache management are separate but optimally run. This work extends from our earlier preliminary work [11] to cover a comprehensive comparative analysis with existing cache invalidation and mobility management approaches in MIPv6 environments. The extension includes identifying conditions under which PICMM performs better than existing schemes, comparing PICMM with a decoupled scheme that optimally but separately manages cache and mobility activities, and analyzing the optimal regional area size as a function of a mobile user's sleep pattern and the server database size. We note that the use of per-user proxies for integrated mobility and service management was discussed in [6]. This work differs from [6] in that it specifically addresses cache consistency issues in query-based client–server mobile applications in MIPv6 environments. By integrating cache consistency management, query management and mobility management based on stateless cache invalidation protocols, it supports frequent MH disconnections and achieves cost minimization.

The rest of the paper is organized as follows. Section 2 describes the system model and mobile database applications, Section 3 describes the proposed integrated cache and mobility management scheme. Section 4 develops a performance model to analyze the cost incurred under the proposed scheme. Section 5 identifies optimal conditions under which our proposed proxy-based algorithm performs the best in terms of network traffic minimization. We compare the performance of our proposed scheme with no-proxy and/or no-caching schemes, as well as with a decoupled scheme that separately manages mobility and service activities, to demonstrate the benefit of integrated cache and mobility management in Mobile IPv6 systems. Finally, Section 6 summarizes the paper and discusses the applicability.

2. Preliminaries

2.1. Querying data objects in mobile client–server applications

We consider mobile client–server applications in Mobile IPv6 environments in which mobile clients query database servers for data objects. To speed up query processing, a client stores a copy of frequently used data objects in its local cache. When a query is issued by the client, the validity of cached data objects accessed by the query is checked first. If these data objects are valid, then the query is immediately processed with very little access time. Otherwise, a fresh copy of invalid data objects will be retrieved to the local store before the query is answered.

2.2. System model

We summarize *dynamic* connection/disconnection, service and mobility characteristics of an MH by several parameters. The first parameter describes the on/off (or wake/sleep) behavior of the MH. We assume that while the MH is in a wake state, it will go to sleep with rate ω_w, and, conversely, while the MH is in a sleep state, it will wake up with rate ω_s.

The second parameter is the residence time that the MH stays in a subnet while it is in a wake state. This parameter can be collected by each MH based on statistical analysis techniques. We assume that future MHs are adequately powerful for collecting data and performing simple statistical analysis. The residence time in general would be characterized by a general distribution. We use the MH's mobility rate (σ) to represent this parameter.

The third parameter is the service traffic between the MH and server applications while the MH is in a wake state. The service rate depends on the query rate for data objects needed by queries, and, if data objects are cached at the MH, the miss ratio. While a query may access several data objects, we assume it is possible to break up a query into subqueries, each accessing a single data object. Thus, it is possible to know a priori the average query rate for a data object i. We characterize the frequency at which the MH accesses data object i by a query rate $\lambda_{q,i}$. Suppose that the MH prefetches N_{data} into its cache. Then the cumulative query arrival rate to access N_{data} data objects, denoted by λ_Q, is given by

$$\lambda_Q = \sum_{i=1}^{N_{\text{data}}} \lambda_{q,i}. \tag{1}$$

The ratio of λ_Q over σ is called the service to mobility ratio (SMR). In general, the SMR of an MH is dynamically changed. The integrated cache and mobility management scheme proposed in this paper allows the optimal regional area to be determined dynamically based on an MH's runtime mobility and service traffic characteristics to minimize the generated network traffic. For efficiency purposes, the MH could build a table to look up its mobility and service rates as a function of its location and time. We assume that the MH will process a query only when it is in a wake state because otherwise the MH would not be able to ascertain if data objects requested are up-to-date.

A query accessing data i stored in an MH's local cache will result in a miss if the data object has been invalidated. Let $P_{\text{miss},i}$ represents the miss ratio for data object i. Then, since an uplink request will be generated by the MH to send to the server only when there is a miss, the overall query rate under cache management to the server is given by

$$\lambda_Q^{\text{cache}} = \sum_{i}^{N_{\text{data}}} \lambda_{q,i} P_{\text{miss},i}. \tag{2}$$

Lastly, the fourth parameter summarizes how often a data object cached at the MH is modified by the server in the mobile application. We use the per data object update rate μ_i to denote this. Assume there are N_{data} data objects cached by the MH in a time window. Assume, for simplicity, that the MH always queries these N_{data} data objects in a time window. When a query accessing a data object results in a miss, we assume a packet is sent from the MH to the server to retrieve a copy of the requested data object. After the data object is received, the query can be answered by the MH. We assume that a reply will take n_D packets to hold a copy of the requested data object.

We measure the communication overhead between two communicating processes by the number of hops in Mobile IPv6 systems, Since the number of subnets separating two communicating processes would not properly measure the hop-count distance, we let $F(K)$ denote a function that maps the number of subnet crossings K to the number of hops. This function $F(K)$ can be periodically and dynamically determined by an MH which collects statistical data as it roams across subnets. An MH would determine this function dynamically, and determine the best service area size to migrate the proxy. In this paper we adopt the fluid flow model [33] assuming that the average number of hops between two communicating processes separated by K subnets is equal to $\sqrt{K}$.

Table 1
Parameters.

Symbol	Meaning
$\lambda_{q,i}$	Query arrival rate for data object i
σ	Mobility rate
μ_i	Data update rate to data object i
ω_w	Disconnection rate to go from awake to asleep
ω_s	Reconnection rate to go from asleep to awake
n_{CT}	Number of packets required for context transfer of *ProxyCache*
n_D	Number of packets to hold a data object
N	Number of server applications engaged
N_{data}	Number of data objects cached at the MH
$F(K)$	A function relating the number of subnets K to the number of hops
K	Number of subnets (or ARs) in a service area
τ	One-hop **round trip** communication delay per packet in wired networks
α	Average distance (in hops) between the HA and the proxy
β	Average distance between the CN and the proxy
γ	Ratio between communication time in wireless networks to that time in wired networks
$P_{\text{miss},i}$	Probability of cache miss for data object i causing an uplink request
P_{wake}	Probability of the MH in the wake state; it is equal to $\omega_s/(\omega_w + \omega_s)$

The system parameters that characterize the mobility and service characteristics of an MH in a Mobile IPv6 system are summarized in Table 1 for easy reference.

3. Integrated cache and mobility management

When an MH starts in a Mobile IPv6 environment, a client-side proxy is created. We assume that access routers (ARs) are powerful and flexible in future all-IP-based wireless networks. Therefore, a client-side proxy can execute on ARs to perform network-layer as well as application-layer functions on behalf of the client, much like a programmable agent in wireless IP systems [7,32].

To improve query performance, the MH may store frequently used data objects in its cache (called *MHCache*). The MH will use invalidation reports received from the corresponding node (CN) through the proxy to validate the content of its cache. We consider that the system operates based on the asynchronous *stateless* strategy, that is, when a data object is changed, the CN immediately sends out an invalidation report to those MHs that keep a cached copy. When a service proxy is created to run on an AR, the proxy is allocated with a buffer space (called *ProxyCache*) to cache invalidation reports and possibly application context sensitive information. The proxy acting on behalf of the MH receives invalidation reports from the CN. If the MH is connected, the proxy forwards them to the MH. If the MH is disconnected, the proxy stores them in *ProxyCache*. Once the MH wakes up, it gets invalidation reports from *ProxyCache* to check if its cache content is current. When the proxy moves because the MH moves across a service area, *ProxyCache* moves with the proxy, thus incurring a *context transfer* cost.

Fig. 1 illustrates our integrated cache and mobility management scheme in Mobile IPv6 environments. When an MH starts in a Mobile IPv6 environment, a client-side proxy is created on the AR, acting as a GFA for the MH. Initially the proxy runs on the first AR of service area 1. The proxy always runs on the first AR in a regional service area as in [29]. When the MH moves within service area 1, the proxy stays put and the MH will inform

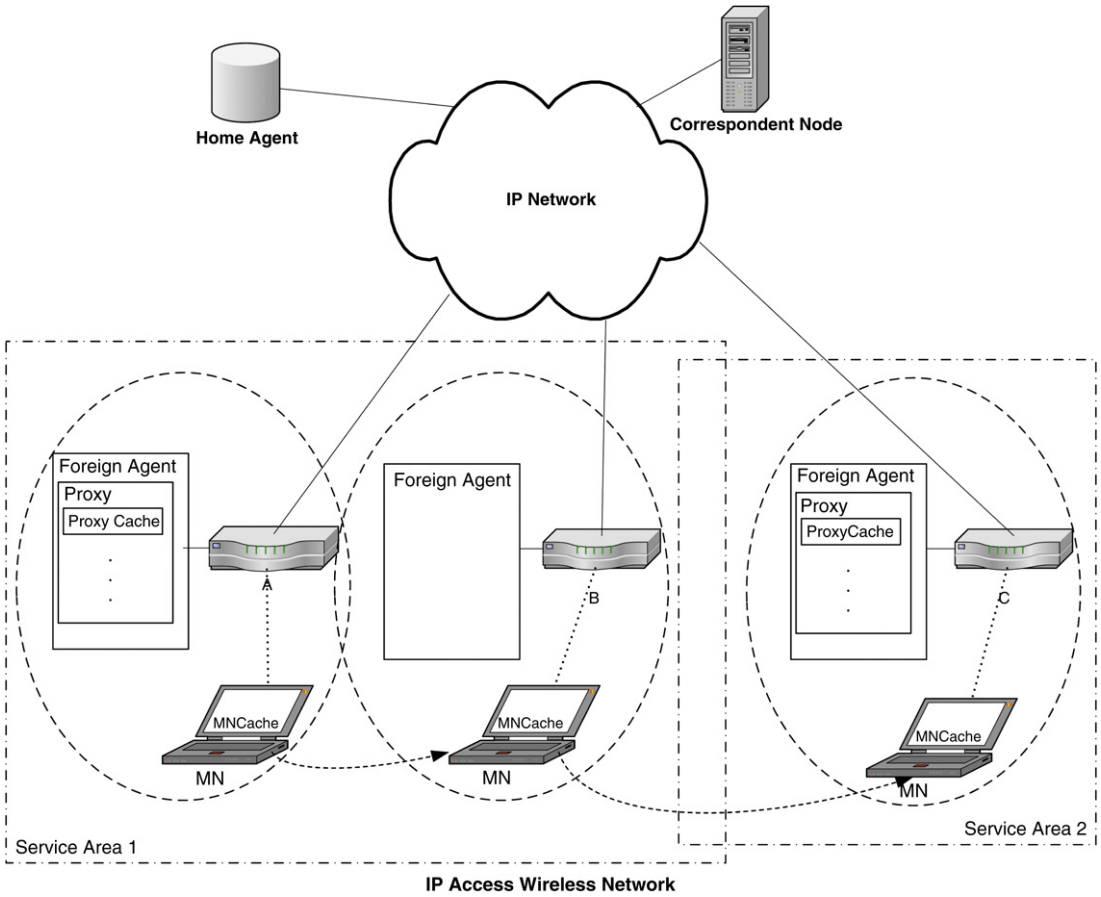

Fig. 1. System infrastructure.

the proxy of its care of address (CoA) change. When the MH moves to service area 2, the proxy also migrates to service area 2 and then runs on the first AR of service area 2. The proxy's *ProxyCache* also moves. In addition, the home agent (HA) and CNs are informed of the CoA change of the proxy. If the MH goes to sleep, its current proxy continues to run to receive invalidation reports. The proxy is destroyed when the MH explicitly issues a disconnection operation. In the case of involuntary disconnection the proxy will also be destroyed after a timeout. A new proxy will be created as the MH restarts if the current proxy cannot be found.

Table 2 gives a pseudo code listing of the PICMM algorithm for processing cache invalidation, query processing, disconnection operations and proxy migration. A trade-off exists to balance the mobility management cost and the cache consistency management cost. A large service area size means that the proxy will not move often, and the query cost due to cache misses could be high because of the triangular routing path CN–Proxy–MH. The cache invalidation cost is also high because of the larger distance between the proxy and the MH when the service area is large. A small service area means that the proxy moves often. This increases the cost of moving ProxyCache with the proxy and to inform the HA and CNs of address change. Thus, an optimal service area size exists. The proxy determines the *dynamic* optimal service area size at runtime. When the MH moves across a subnet boundary, the proxy will check if the service area is crossed. If yes, a new optimal service area size is determined by executing a computational procedure developed in this paper based on the MH's mobility and service characteristics in the new service area. Fig. 2 illustrates the proxy migration process as a result of service area boundary crossing during an ongoing service session with a CN.

4. Performance model and parameterization

4.1. Model

We develop a performance model based on Stochastic Petri nets to analyze the cache management cost and mobility management cost incurred due to the employment of the integrated mobility and cache management scheme. The objective is to derive an equation from the analytical model to allow the overall network traffic cost incurred to be calculated as a function of the number of subnets covered (K) in a service area, when given a set of parameters as input to characterize a MH's mobility and service behaviors, including the mobility rate (σ), query arrival rate ($\lambda_{q,i}$), disconnection and reconnection rates (ω_s and ω_w), and data object update rate μ_i. This computational procedure allows us to determine the optimal service area size (K_{opt}) to be deployed at runtime to minimize the overall network traffic cost due to cache and mobility management. We choose SPN because of its ability to deal with general time distributions for events, its concise representation of the underlying state machine to deal with a large number of states, and its expressiveness to reason about an MH's behavior as it migrates from one state to another in response to events occurring in the system.

Fig. 3 shows the performance model based on Stochastic Petri nets. Table 3 gives the meaning of places and transitions defined in the model.

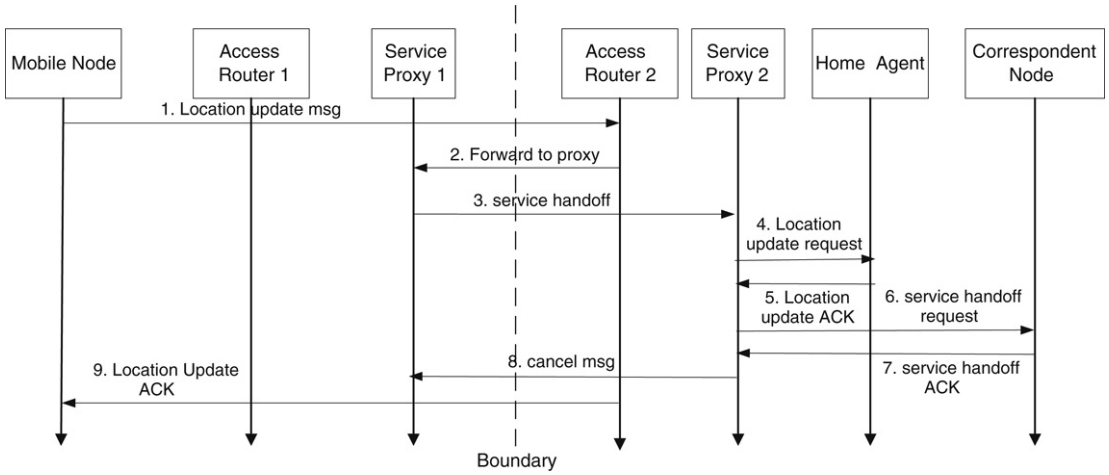

Fig. 2. Service handoff process when crossing a service area by an MH.

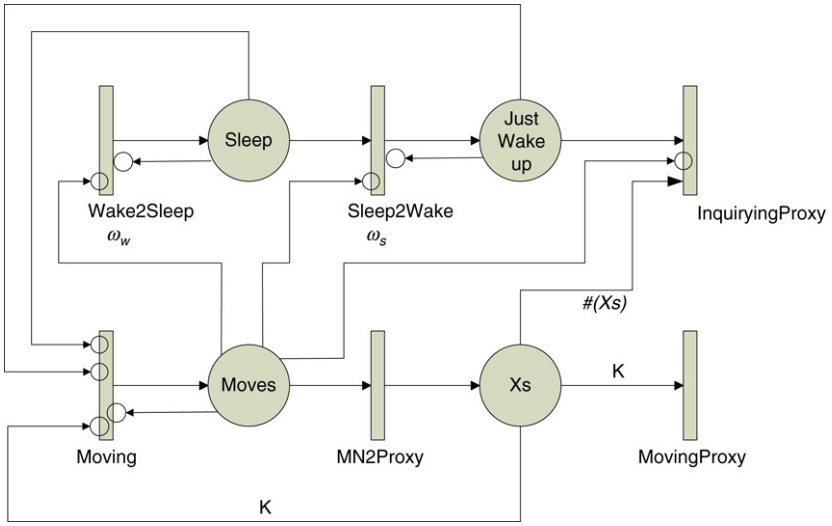

Fig. 3. Petri net model.

We first briefly introduce the nomenclature necessary for the comprehension of an SPN model. An SPN model consists of entities including transitions, places, arcs, and tokens. A token is used as a marker; it can be used to represent an entity such as a user or a task. A place is a token place-holder; it is normally given a distinct name that conveys the meaning of a state component, e.g., place `Sleep` in Fig. 2 means that the MH is in sleep mode. The function Mark(P) returns the number of tokens in place P. Typically, places in the SPN model correspond to state components in the underlying Markov or semi-Markov model. A transition represents an event. If a timed transition is fired in an SPN then it means that an event associated with the transition has occurred, e.g., a subnet cross event occurs after a time exponentially distributed (or generally distributed) with parameter σ has elapsed in the SPN model. Arcs connect places to transitions. We differentiate input arcs from output arcs. An input arc goes from an input place to a transition, while an output arc goes from a transition to an output place. Input arcs can be further distinguished by regular input arcs (with arrow as the notation) vs. inhibitor input arcs (with arrow ended with a circle as the notation). An arc can be associated with a multiplicity to indicate the number of tokens associated with the arc; the default is 1 if not specified. A transition can fire if the following two conditions are satisfied: (a) there are at least m tokens in each input place connected to it by an input arc with multiplicity of m; (b) there are less than n tokens in each input place connected to it by an inhibitor arc with multiplicity of n. For example, `InquiryingProxy` can fire if there is at least one token in place `Just Wake up` and no token at place `Moves`. It does not require `Xs` to contain any token because the input arc multiplicity from `Xs` is defined as the number of tokens held in `Xs`.

In Fig. 3, the number of tokens accumulated in place `Xs`, that is, Mark(`Xs`), represents the number of subnets crossed by the MH since the MH enters a new service area. We allow it to accumulate to K, at which point we perform a service handoff. We construct the SPN model to describe the behavior of an MH operating under our proposed integrated cache and mobility management scheme. By varying K, the SPN model allows us to compute the cache and mobility management cost as a function of K, thereby allowing the optimal K to be determined. Below we describe how we construct the SPN model:

- When an MH moves across a subnet area, a token is put in place `Moves`. The mobility rate at which location handoffs occur is σ, which is the transition rate assigned to `Moving`.
- After moving into a subnet, the MH obtains a new CoA and informs the proxy (that acts as a GFA) of the CoA change. This is modeled by enabling and firing transition `MH2Proxy` while disabling transition `Moving`. After `MH2Proxy` is fired, a token in place `Moves` flows to place `Xs`, representing that a location handoff has been completed and the proxy has been informed of the CoA address change of the MH. The rate at which `MH2Proxy`

Table 2
Algorithm description.

Proxy Set-Up: **When** the MH starts or restarts after sleep in a Mobile IPv6 environment **if** the current client-side proxy does not exist **then** a new client-side proxy is created for the MH to run the current AR.
Proxy Tear-Down: **When** the MH explicitly issues a disconnection or is inactive after a timeout the proxy is destroyed.
Cache Invalidation: On receiving an invalidation report **if** the MH is connected to the wireless network **then** the proxy forwards the invalidation report to the MH; **else** {the proxy stores the invalidation report in ProxyCache; when the MH wakes up, the proxy forwards the invalidation report.}
Query Processing: On receiving a query from the user **if** the data object requested is in the MH's cache and valid **then** the MH returns the cached object in the query result; **else** {there is a cache miss; user request is sent to the proxy; the proxy forwards the request to the CN; the CN returns a copy of the requested object to the proxy; the proxy forwards the data object to MH; the MH stores the data object in MHCache; the MH returns the query result.}
Reconnection after a Sleep: **if** the MH is in the same subnet as the proxy **then** the MH gets invalidation reports from ProxyCache; **else** {the proxy moves to MH's current subnet; ProxyCache is moved with the proxy; the proxy informs the HA and CNs of its CoA change; the MH gets invalidation reports from ProxyCache. }
Proxy Migration: **if** the MH moves to a new service area The MH updates its location information with the proxy through the AR it just moves into (AR2 shown in Fig. 2). When the proxy realizes that a service area has been crossed, it initiates a migration process to move to AR2. Once the proxy moves into the new service area and runs on AR2, it informs the HA and the CN of its new address.

Table 3
Meanings of places and transitions in the SPN model.

Symbol	Meanings
`Sleep`	Mark(`Sleep`) = 1 means the MH just entered sleep mode.
`Wake2Sleep`	A timed transition to change from awake to sleep.
`Just Wake up`	Mark(`Just Wake up`) = 1 means the MH just woke up.
`Sleep2wake`	A timed transition to change from sleep to awake.
`InquiryingProxy`	A timed transition to query the proxy.
`Moves`	Mark(`Moves`) = 1 means that the MH has just moved across a subnet area.
`Moving`	A timed transition to move across a subnet area.
`MH2Proxy`	A timed transition for the MH to communicate with the proxy for an intra-regional move
`Xs`	Mark(`Xs`) indicates the number of subnets that have been crossed by the MH.
`MovingProxy`	A timed transition for proxy to move to a new server area.

is fired depends on the number of subnets separating the MH and the proxy.

- If the number of tokens in place `Xs` has accumulated to K, a threshold representing the size of a service area, then it means that the MH has just moved into a new service area and a service handoff ensues. This is modeled by assigning an enabling function that will enable transition `MovingProxy` after K tokens have been accumulated in place `Xs`. After transition `MovingProxy` is fired, all K tokens are consumed and place `Xs` contains no token, representing the action that the proxy has just moved into a new service area. The rate at which transition `MovingProxy` fires depends on the cost of informing the HA and CNs of the proxy CoA change and the cost of transferring *ProxyCache* to the new proxy location.
- The MH alternates between "sleep" and "wake" states due to connection and disconnection. Initially the MH is in the wake state. After a time is elapsed representing the online connection time, the MH goes to the sleep state. This is modeled by having a token flow into place `sleep`. The transition rate to transition `wake2Sleep` is ω_s. Note that if the MH is already in place `sleep`, transition `wake2Sleep` cannot fire.
- While the MH is in sleep mode, after a time is elapsed representing the disconnection time, the MH goes to the wake state. This is modeled by having a token flow from place `wake2Sleep` to place `just-wakeup`. The transition rate to transition `sleep2Wake` is ω_w. If the MH is already in the wake state, transition `sleep2Wake` cannot fire.
- After the MH wakes up, it will check with the proxy of the status of its cached data objects. If the proxy is not in the same subnet, then the proxy moves to the subnet that the MH currently resides. The cost involved includes the cost of service context transfer of *ProxyCache* and the signaling cost of informing the HA and CNs of the address change. This is modeled by firing the `Proxy` transition with a transition rate reflecting the cost (see how we do parameterization below). Firing this transition will flush all the tokens in place `Xs` as if a service handoff had happened. This is modeled by a variable input arc from place `Xs` to transition `Proxy`.
- While the MH is in state `sleep`, it will not move, so we use an inhibitor arc from place `sleep` to transition `Moving` to prohibit transition `Moving` from firing. Similarly we use an inhibitor arc from place `just-wakeup` to transition `Moves`, and an inhibitor arc from place `Moves` to all other transitions.

4.2. Parameterization

Below we describe how we parameterize the SPN model. Transitions `Moving`, `wake2Sleep` and `sleep2Wake` have transition rates of σ, ω_w and ω_s, respectively. The transition rates of the three remaining transitions need to be parameterized (i.e., given values to) given basic parameter values. The firing time of transition `MH2Proxy` stands for the communication time of the MH registering with the proxy through the wireless network. This time depends on the number of hops separating the MH and its proxy. Thus, the transition rate of transition `MH2Proxy` is calculated as

$$\frac{1}{\gamma\tau + F(\text{Mark}(Xs) + 1) \times \tau}$$

where τ stands for the one-hop communication delay per packet in the wired network and γ is a proportionality constant representing the ratio of the communication delay in the wireless network to the communication delay in the wired network. $F(\text{Mark}(\texttt{Xs}) + 1)$ returns the number of hops between the current subnet and the proxy separated by $\text{Mark}(\texttt{Xs}) + 1$ subnets. The argument of the $F(x)$ function is added by 1 to satisfy the initial condition that $\text{Mark}(\texttt{Xs}) = 0$ in which the proxy has just moved into a new service area, so at the first subnet crossing event the distance between the proxy and the subnet is one subnet apart. Note that this transition rate is state dependent because the number of tokens in place `Xs` changes dynamically over time.

When transition `MovingProxy` fires, the proxy will move into a service area after invoking a service handoff. The cost involved includes informing the HA and CNs of the CoA address change, and transferring the service context information stored in *ProxyCache*. The transition rate of transition `MovingProxy` thus is calculated as

$$\frac{1}{n_{CT}F(K)\tau + (\alpha + N\beta)\tau}$$

where $F(K)$ returns the number of hops for two subnets separated by K subnets, n_{CT} is the number of packets required to carry the service context information during a proxy transfer, α is the average distance between the proxy and the HA, N is the number of server applications (or CNs) which the MH engages concurrently, and β is the average distance between the proxy and a CN. The proxy could determine the values of α and β based on statistical data collected on the fly.

When transition `InquiryingProxy` fires, the MH will contact the proxy. If the proxy is in the same subnet as the current MH resides, the cost is the transmission delay from the AR to the MH, that is, $\gamma\tau$. If the proxy is not in the same subnet, the cost includes the delay for contacting the proxy, moving the proxy to the current AR, and informing the HA and CNs of the proxy's CoA address change. Thus, the transition rate of transition *InquiryingProxy* is calculated as

$$\begin{cases} \dfrac{1}{\gamma\tau} & \text{if Mark}(Xs) = 0; \\ \dfrac{1}{F(\text{Mark}(Xs))\tau + n_{CT}F(\text{Mark}(Xs))\tau + (\alpha + N\beta + \gamma)\tau} & \text{if Mark}(Xs) > 0. \end{cases}$$

4.3. Performance metric and cost function

4.3.1. Metric

An MH and its proxy determine the service area dynamically to minimize the overall cache and mobility management network traffic cost incurred by the MH. There are three costs to be considered: a cost for forwarding a user query uplink to the server to obtain a copy of the data object because of cache miss, a cost for forwarding invalidation reports from the proxy to the MH, and a cost for mobility management, including handling location and service handoffs. The overall network traffic cost, i.e., the performance metric that we aim to minimize, is the sum of these three costs *per time unit*. Note that this performance metric gives the network traffic incurred per time unit per MH, so even a 5%–10% performance improvement could impact the IP network performance significantly because the cumulative cost over a long period of time for all MHs would be very significant.

4.3.2. Cost function derivation

Let C_{query} be the average communication cost to service a query. Let C_{mobility} be the average communication cost to service a location handoff, including one that can trigger a service handoff. Let $C_{\text{invalidation}}$ be the average communication cost to forward an invalidation report. Finally, let C_{total} be the overall cost incurred *per time unit*. Then, C_{total} is the sum of the product of the respective communication cost multiplied with the rate at which the respective event occurs, that is,

$$C_{\text{total}} = \lambda_e \times C_{\text{query}} + \sigma_e \times C_{\text{mobility}} + \mu_e \times C_{\text{invalidation}}. \quad (3)$$

Here λ_e is the effective data query rate, σ_e is the effective mobility rate, and μ_e is the effective data update rate.

The effective query arrival rate λ_e is the query arrival rate multiplied with the probability of the MH being awake because queries are issued only when the MH is awake:

$$\lambda_e = \lambda_Q \times P_{\text{wake}} \quad (4)$$

where λ_Q is the aggregate query arrival rate as given in Eq. (1) and P_{wake} is the probability of the MH being in the wake state, given by

$$P_{\text{wake}} = \frac{\omega_s}{\omega_s + \omega_w}. \quad (5)$$

The effective mobility rate σ_e is the mobility rate multiplied with the probability of the MH being awake again because the MH does not trigger mobility handoff events while it is in sleep mode. Thus,

$$\sigma_e = \sigma \times P_{\text{wake}}. \quad (6)$$

On the other hand, the effective data update rate μ_e is simply the aggregate data update rate, calculated as

$$\mu_e = \sum_{i=1}^{N_{\text{data}}} \mu_i. \quad (7)$$

Here the expression for μ_e is not multiplied with the probability of the MH being awake because updates to data occur regardless of whether the MH is in sleep or wakeup mode.

Below we derive C_{query}, C_{mobility} and $C_{\text{invalidation}}$. The stochastic model underlying the SPN model is a continuous-time semi-Markov chain[1] with the state representation of (a, b, c, d) where a is the number of tokens in place `Moves`, b is the number of tokens in place `Xs`, c is the number of tokens in place `Sleep`, and d is the number of tokens in place `Just Wake Up`. The distribution of tokens in these four places makes up the states of the system. In general let state i represent a particular state represented by (a, b, c, d). Let P_i be the steady state probability that the system is in state i. The P_i's can be obtained by applying numerical analysis methods such as SOR or Gauss Seidel to solve the underlying model.

Let $C_{i,\text{query}}$ be the communication cost for answering a query given that the MH is in state i. Then, C_{query} can be calculated as a weighted average of $C_{i,\text{query}}$'s as follows:

$$C_{\text{query}} = \sum_i (P_i \times C_{i,\text{query}}). \quad (8)$$

Deriving $C_{i,\text{query}}$ requires knowledge of $P_{\text{miss},j}$, i.e., the probability of cache miss of data object j, because this cost depends on whether the requested cached data object is up-to-date and can be used to answer a query. A cache miss happens if an update has occurred to object j prior to the query requesting object j arriving at the MH. Inevitably this depends on the relative magnitudes of $\lambda_{q,j}$, μ_j and ω_w and ω_s. The calculation of $P_{\text{miss},j}$ essentially hinges on the competition between the *effective query arrival rate* and the update rate (with respect to time), since the cached data object will be invalidated when an update occurs before a query arrives. Since the MH will not issue queries while it is in sleep mode, the effective query arrival rate for object j is equal to the query arrival rate for data object j multiplied with the probability of being awake, i.e., $P_{\text{wake}}\lambda_{q,j}$, or $[\omega_s/(\omega_s + \omega_w)]\lambda_{q,j}$. Thus, $P_{\text{miss},j}$ is calculated as

$$P_{\text{miss},j} = \frac{\mu_j}{(\mu_j + [\omega_s/(\omega_s + \omega_w)]\lambda_{q,j})}. \quad (9)$$

Suppose that a query or a subquery is asking for data object j. If data object j being queried is in the MH's cache and is valid, then there is a cache hit and the query cost is zero. Otherwise, there is a cache miss, and the query cost will include a communication cost between the MH and the proxy, and a cost from the proxy to the CN. Thus, the average query cost for a query asking for data object j when the MH is connected is given by $(\gamma\tau + \beta\tau + F(\text{Mark}(Xs))\tau) \times n_D \times P_{\text{miss},j}$.

On the other hand, when the MH just wakes up, i.e., in state "just wake up", the MH will first check with the proxy regarding the cache status when answering a query, and, if the cached data object is not valid, will get a copy from the server. Thus, the cost is

[1] If the elapsed time of a timed transition is generally distributed then the underlying model is a semi-Markov chain; if the elapsed time is exponentially distributed, then the underlying model is a Markov chain.

$\gamma\tau + (\gamma\tau + \beta\tau + F(\text{Mark}(Xs))\tau) \times n_D \times P_{\text{miss},j}$, where the first term is the cost for the MH to get invalidation reports from the proxy (which has moved to local) to check the cache status and the second term is for the cost to get a copy of object j from the CN if there is a miss. Lastly, when the MH is in sleep mode, there is no query cost because no query is issued while the MH is in sleep mode. Thus,

$$C_{i,\text{query}} = \begin{cases} 0 & \text{if Mark(Sleep)} > 0 \\ \gamma\tau + (\gamma\tau + \beta\tau + F(\text{Mark}(Xs))\tau) n_D P_{\text{miss},j} & \\ \quad \text{else if Mark(Just Wake Up)} > 0 & \\ (\gamma\tau + \beta\tau + F(\text{Mark}(Xs))\tau) \times n_D \times P_{\text{miss},j} & \text{otherwise.} \end{cases}$$

Let $C_{i,\text{mobility}}$ be the communication cost to service a location handoff given that the MH is in state i. C_{mobility} is calculated as a weighted average as follows:

$$C_{\text{mobility}} = \sum_i (P_i \times C_{i,\text{mobility}}). \tag{10}$$

If in state i, $\text{Mark}(Xs) < K$, then the MH will only inform the proxy of the CoA address change. On the other hand, if $\text{Mark}(Xs) = K$, then the location handoff also triggers a service handoff. A service handoff will incur a context transfer cost of *ProxyCache* while moving the proxy to the new service area and a communication cost to inform the HA and N CNs (or application servers) of the CoA address change of the proxy. When the MH is in the sleep state, there is no location handoff cost since the MH is disconnected from the system. When the MH just wakes up, it will look for its proxy. If the proxy is in the same subnet as MH currently resides because the MH does not move during sleep, the cost involved is only for contacting the current AR for the proxy location. Otherwise, the proxy is moved to the current subnet, and the cost is that of a service handoff, i.e., a context transfer cost and a cost to inform the HA and N CNs of the CoA address change of the proxy. Therefore,

$$C_{i,\text{mobility}} = \begin{cases} 0 & \text{if Mark(Sleep)} > 0 \\ \gamma\tau & \text{else if Mark(Just Wake Up)} > 0 \text{ and Mark(Xs)} = 0 \\ \gamma\tau + \alpha\tau + N\beta\tau + F(\text{Mark}(Xs))n_{CT}\tau & \\ \quad \text{else if Mark(Just Wake Up)} > 0 \text{ and Mark(Xs)} > 0 & \\ \gamma\tau + F(\text{Mark}(Xs))\tau & \text{else if Mark}(Xs) < K \\ \gamma\tau + \alpha\tau + N\beta\tau + F(K)n_{CT}\tau & \text{else if Mark}(Xs) = K. \end{cases}$$

Lastly, let $C_{i,\text{invalidation}}$ be the cost to forward an invalidation report from the proxy to the MH when the MH is in state i. Similarly, the weighted cost per invalidation report is calculated as

$$C_{\text{invalidation}} = \sum_i (P_i \times C_{i,\text{invalidation}}). \tag{11}$$

When an update occurs, the CN sends an invalidation report to the MH. If the MH is in sleep or just wakeup mode, then the invalidation report is buffered in the proxy, so the cost is from the CN to the proxy. Otherwise the cost is from the CN through the proxy to the MH. Thus,

$$C_{i,\text{invalidation}} = \begin{cases} \beta\tau & \text{if Mark(Sleep)} > 0 \text{ or Mark(Just Wake Up)} > 0 \\ \beta\tau + F(\text{Mark}(Xs))\tau + \gamma\tau & \text{otherwise.} \end{cases}$$

Summarizing the above, Eq. (3) along with other formulas derived in this section allows one to calculate the network traffic incurred per time unit for an MH characterized by a set of parameter values.

Table 4
Parameters and default values used in performance analysis.

Parameters	Default value
τ	0.025 s
N	1
n_{CT}	2 packets
n_D	1 packet
α	30 hops
β	30 hops
γ	5
N_{data}	Range {1, 10, 50, 100, 200, 500} (objects)
$\lambda_{q,i}$	Range {1/100, 1/50, 1/10, 1/5, 1} (number of times per second)
σ	Range {0.005, 0.01, 0.05, 0.1, 0.15} (number of times per second)
μ_i	Range {1/500, 1/100, 1/50, 1/10} (number of times per second)
ω_w/ω_s	Range {1/16, 1/8, 1/4, 1/2, 1.0, 2.0, 4.0, 8.0}

5. Performance evaluation

In our proposed scheme, an MH and its proxy would apply Eqs. (3), (8), (10) and (11) to calculate C_{total} as a function of K and determine the optimal K representing the optimal *service area* size that will minimize the network signaling cost.

To provide a better sense of the performance improvement of our proposed proxy-based scheme for integrated cache and mobility management, we compare our scheme with three baseline schemes: a *no-proxy no-caching* (NPNC) scheme, a *proxy no-caching* (PNC) scheme, and a *no-proxy caching* (NPC) management scheme. We also compare our scheme with a decoupled scheme that optimally but separately manages mobility and service activities. The NPNC scheme essentially is the basic MIPv6 scheme without using a proxy for either mobility or cache management. The PNC scheme is the proxy-based regional registration scheme using a proxy for mobility management. In these two schemes, the MH does not cache data objects. The NPC scheme uses basic MIPv6 for mobility management and cached data objects maintained by the MH for cache management, but there is no proxy being used.

The set of parameters along with their default values used in the performance study is given in Table 4. The physical meanings of these parameters are given in Table 1. We analyze the impact of key parameters by varying their values and observing their effects on C_{total} and K_{opt}.

Below we explain the default parameter values used and how we may obtain the parameter values in real time. It is reported that the average delay of a wired link is about 20 to 50 ms [2]. Therefore the one-hop communication delay in the wired network, τ, is set to 25 ms. On the other hand, it is reported that the average delay of a wireless link is about 200 ms [28]. Thus, the ratio of the communication delay in the wireless network to the communication delay in the wired network, γ, is set to 5. The number of server applications that the MH is concurrently engaged is application dependent. We set $N = 1$. The number of packets required for migrating the service context of the proxy should be minimized for scalability. We set $n_{CT} = 2$, with one packet for holding cache information and one packet for holding mobility information. The average distance (in hops) between the HA and the proxy, α, may be measured by the proxy by examining packets routed through in run time. We assume it is 30 in the performance analysis. Similarly we assume that the average distance between the CN and the proxy, β, is also 30. The number of data objects cached by the MH is also application dependent. We vary N_{data} in the range of {1, 10, 50, 100, 200, 500} to analyze its effect. The query arrival rate for object i, $\lambda_{q,i}$, is in the range of {1/100, 1/50, 1/10, 1/5, 1.0} to account for the possibility of low to high query scenarios, ranging from once per 100 s to once per second. The MH mobility rate, σ, is in the range of {0.005, 0.01, 0.05, 0.1, 0.15} to account for low to high mobility behaviors, ranging from once

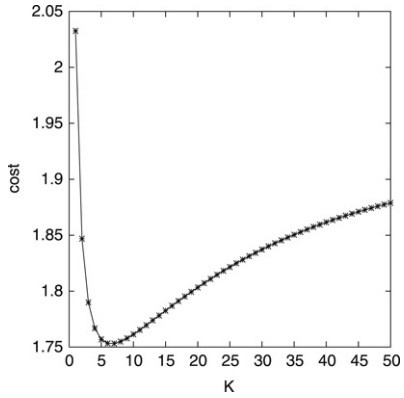

Fig. 4. Cost vs. K.

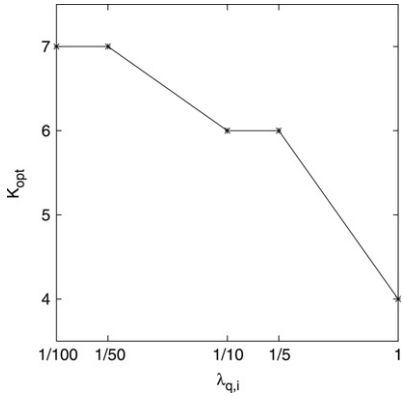

Fig. 5. K_{opt} vs. $\lambda_{q,i}$.

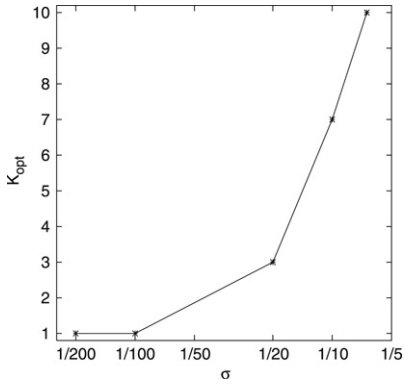

Fig. 6. K_{opt} vs. σ.

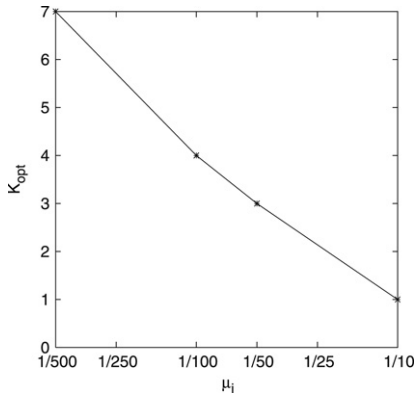

Fig. 7. K_{opt} vs. μ_i.

(i.e., one subnet boundary crossing) per 200 s (by walk) to once per 2/3 of a second (by car). The update rate for data object i, μ_i, is in the range of 1/500, 1/100, 1/50, 1/10 to account for low to high object update activities, from once per 500 s to once per 10 s. Finally, the MH's sleep to wake ratio, ω_w/ω_s, is in the range of {1/16, 1/8, 1/4, 1/2, 1.0, 2.0, 4.0, 8.0} to account for the sleep behaviors, ranging from workaholics to sleepers. Without loss of generality, we consider the case that $\lambda_{q,i}$ is the same for all N_{data} objects, and that μ_i is the same for all cached data objects.

The numerical data reported are obtained based on Eqs. (3), (8), (10) and (11). For ease of exposition, we normalize C_{total} with respect to $\tau = 1$ (that is, the per-hop cost is set to (1)), so the cost corresponds to the number of hops that packets need to travel per second. Note that the cost is on a per-sec and per-MH basis so the cumulative cost saved for all MHs over time is significant. The goals of the numerical analysis are to (a) show that there exists an optimal service area under our cache scheme for network traffic cost minimization in Mobile IPv6 systems; (b) compare our integrated cache and mobility management scheme with existing schemes to demonstrate its benefit and identify conditions under which our scheme performs the best; and (c) study the effect of model parameters, including the query arrival rate $\lambda_{q,i}$, mobility rate σ, sleep pattern ω_w/ω_s and data update rate μ_i, on the optimal service area size.

5.1. Optimal service area for integrated cache and mobility management in MIPv6

Fig. 4 shows that under our integrated scheme there exists an optimal proxy service area size K_{opt} to minimize the overall network traffic cost, when given a set of parameter values characterizing the mobility and service behaviors of the MH in Mobile IP networks.

We observe from Fig. 5 that K_{opt} exists for all $\lambda_{q,i}$ values. We see that K_{opt} decreases slowly as $\lambda_{q,i}$ increases. The reason is that as $\lambda_{q,i}$ increases, the query cost increases, and subsequently the MH prefers a small service area size to reduce the query cost.

We observe from Fig. 6 that K_{opt} increases as σ increases. The reason is that when the mobility rate is high, the mobility management cost is also high. Therefore, the proxy likes to stay at a large service area to reduce the location handoff cost such that a location handoff will most likely only involve informing the proxy of the location change without incurring a service handoff to migrate the proxy. As a result, when σ increases, K_{opt} increases.

Fig. 7 shows that K_{opt} decreases as μ_i increases. The reasons is that as μ_i increases, the data in the local cache are more likely to be out-of-date. The MH in this case will more likely to send queries to the server to obtain the latest version of data through the proxy. Also more invalidation reports will be sent from the CN to the MH through the proxy. Therefore, the MH will stay close to the proxy to reduce the triangular CN–proxy–MH communication cost.

Fig. 8 shows the relationship between K_{opt} vs. the sleep ratio. We observe that K_{opt} decreases as the MH sleeps longer. The reason is that the data in MH's local cache are more likely to be out-of-date when the MH sleeps longer. Consequently, the MH will stay close to the proxy to reduce the triangular CN–proxy–MH communication cost for sending the query to and receiving replies from the server.

Fig. 9 shows the relationship between K_{opt} vs. cache size. We observe that K_{opt} decreases as the number of cached data objects increases. The reason is that as the number of cached data objects increases, more invalidation reports will be sent from the CN to the MH, given the same update rate for all data objects. In order to reduce the triangular CN–proxy–MH cost for routing invalidation reports, the MH tends to stay closer to the proxy.

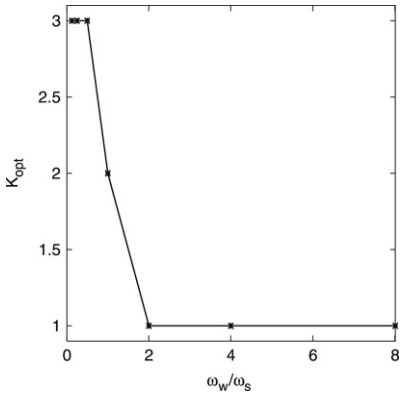

Fig. 8. K_{opt} vs. ω_w/ω_s.

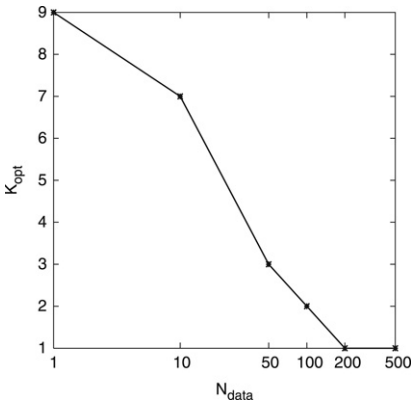

Fig. 9. K_{opt} vs. N_{data}.

5.2. Performance comparison

Here we compare our proxy-based integrated cache and mobility management (PICMM) scheme with the no-proxy caching (NPC), proxy no-caching (PNC) and no-proxy no-caching (NPNC) management schemes. The NPNC scheme corresponds to the basic MIPv6 scheme. Here we emphasize that the performance metric, the total traffic incurred per time unit per MH, is cumulative in nature, so a performance improvement of 5%–10% is considered significant since the cost accumulated over a long period covering all MHs would be significant.

We first derive the total cost incurred *per time unit* for these baseline schemes. Let $C_{\text{total\_NPC}}$ be the total cost incurred *per time unit* under the no-proxy caching scheme in Mobile IP networks. Similar to our integrated scheme, $C_{\text{total\_NPC}}$ can be computed by

$$C_{\text{total\_NPC}} = \lambda_e \times C_{\text{query\_NPC}} + \sigma_e \times C_{\text{mobility\_NPC}} + \mu_e \times C_{\text{invalidation\_NPC}}$$

where λ_e, σ_e and μ_e are defined as before by Eqs. (4), (6) and (7), respectively. For the query cost, the MH will check with the CN when there is a cache miss. Thus,

$$C_{\text{query\_NPC}} = (\beta\tau + \gamma\tau) \times P_{\text{miss},j}.$$

For the mobility cost, when the MH triggers a location handoff, it will inform the HA and CNs. Thus,

$$C_{\text{mobility\_NPC}} = \alpha\tau + N\beta\tau + \gamma\tau.$$

For the invalidation cost, the CN will always inform the MH whenever there is an update. Thus,

$$C_{\text{invalidation\_NPC}} = \beta\tau + \gamma\tau.$$

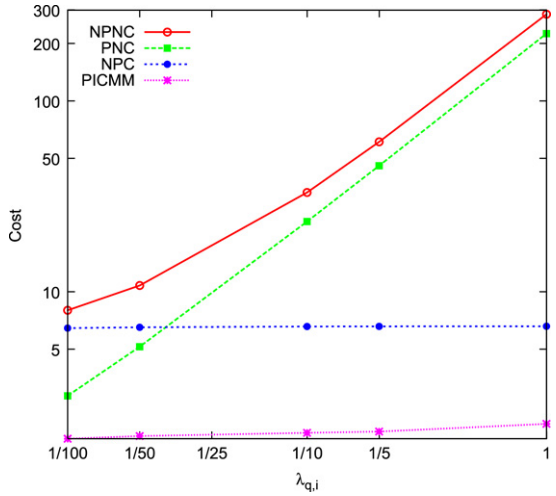

Fig. 10. Generated network traffic as a function of $\lambda_{q,i}$.

For the proxy no-caching (PNC) scheme, let $C_{\text{total\_PNC}}$ be the total cost incurred *per time unit*. Then, $C_{\text{total\_PNC}}$ can be computed by

$$C_{\text{total\_PNC}} = \lambda_e \times C_{\text{query\_PNC}} + \sigma_e \times C_{\text{mobility}} \quad (12)$$

where C_{mobility} is computed based on Eq. (10) as before, and $C_{\text{query\_PNC}}$ is computed much the same way as C_{query} is computed in Eq. (8), except that if the MH is awake, the MH always sends the query to the CN through its proxy since the MH does not cache data objects. Thus,

$$C_{i,\text{query\_PNC}} = \begin{cases} 0 & \text{if Mark(Sleep) or Mark(Just Wake Up)} > 0 \\ \gamma\tau + \beta\tau + F(\text{Mark}(Xs))\tau & \text{otherwise.} \end{cases}$$

Note that there is no cache invalidation cost under PNC since there is no cache used.

Finally for the no-proxy no-caching scheme (i.e., basic MIPv6), let $C_{\text{total\_NPNC}}$ be the total cost incurred *per time unit*. Then $C_{\text{total\_NPNC}}$ can be computed as

$$C_{\text{total\_NPNC}} = \lambda_e \times C_{\text{query\_NPNC}} + \sigma_e \times C_{\text{mobility\_NPNC}}$$

where $C_{\text{query\_NPNC}}$ is the cost from the MH to the CN, i.e., $\beta\tau + \gamma\tau$ on average, and $C_{\text{mobility\_NPNC}}$ is the cost to inform the HA and CNs, computed as $\alpha\tau + N\beta\tau + \gamma\tau$ on average.

Fig. 10 compares our proposed PICMM scheme vs. NPNC, PNC and NPC management schemes in the network traffic cost generated, as a function of $\lambda_{q,i}$ with all other parameters fixed. For the PNC scheme, the cost shown is the minimized network traffic generated at the optimal service area. From Fig. 10 we see that caching-based schemes (NPC and PICMM) achieve much better performance for non-caching-based schemes (NPNC and PNC), especially when $\lambda_{q,i}$ is large. The cost of either NPC or PICMM increases slowly as $\lambda_{q,i}$ increases, while the cost for either PNC or NPNC increases drastically as $\lambda_{q,i}$ increases. The reason is that at high $\lambda_{q,i}$ values, the dominating network traffic is due to sending queries to the server. Caching-based schemes can greatly reduce the magnitude of server-querying network traffic because caching allows some of the queries to be processed by the MH based on up-to-date cached data objects. This trend is true when the update rate to cached data objects (μ_i) is low to modest so that the cost saving due to query processing for sending queries to the server outweighs the extra cost due to triangular routing for receiving invalidation reports and query replies. On the other hand, between the two caching-based schemes, our PICMM scheme performs consistently better than NPC due to the use of a proxy for integrated cache and mobility management for extra cost saving.

Fig. 11 compares our proposed PICMM scheme vs. NPNC, PNC and NPC management schemes in the generated network

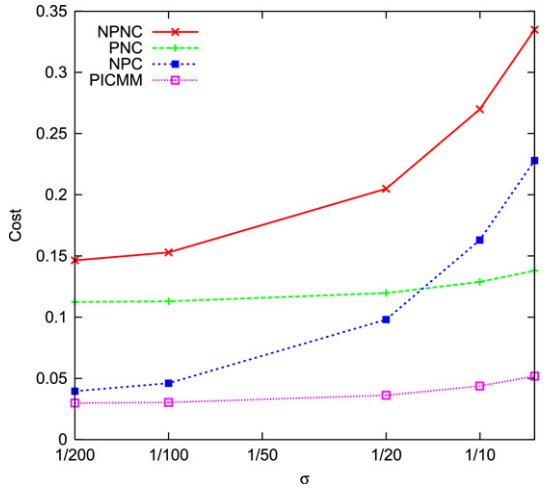

Fig. 11. Generated network traffic as a function of σ.

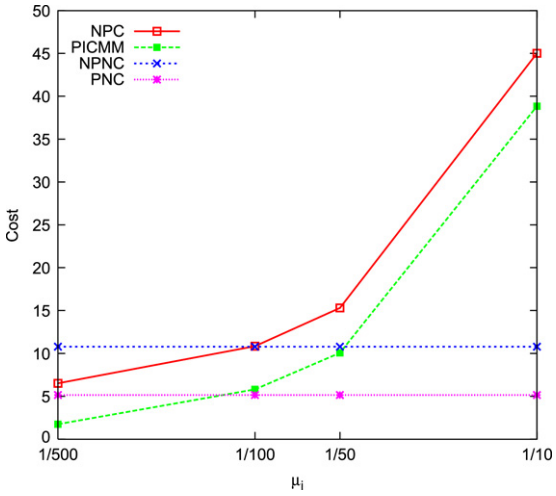

Fig. 13. Generated network traffic as a function of μ_i.

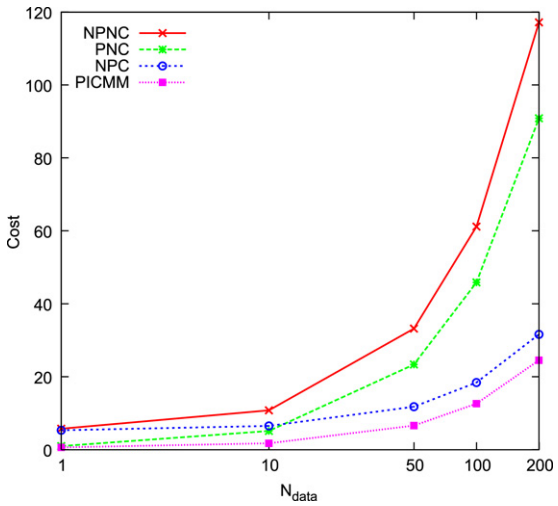

Fig. 12. Generated network traffic as a function of cache size.

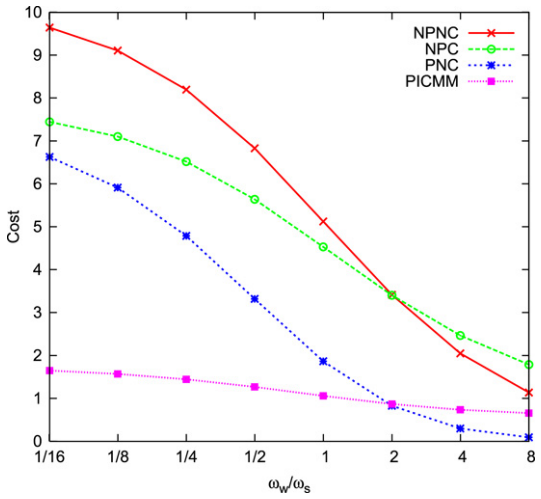

Fig. 14. Generated network traffic as a function of ω_w/ω_s.

traffic, as a function of σ with all other parameters fixed. Fig. 11 shows that our proposed PICMM scheme outperforms all other schemes. Compared with the NPNC scheme, the performance of our scheme is especially pronounced when σ is high. The reason is that PICMM uses a proxy to serve as a GFA to reduce the cost of location handoffs and the benefit is especially pronounced when the mobility rate of the MH is high. Compared with the NPC scheme, PICMM considers integrated cache management and mobility management. Consequently, the best service area determined can minimize the overall network traffic cost better than that by NPC which runs mobility management independently of cache management.

In Fig. 12 we compare PICMM vs. other schemes in the generated network traffic, as a function of the cache size. The total cost incurred under PICMM increases as the number of cached data items increases. The main reason is that the invalidation cost increases as the number of cached data items increases. Compared with non-caching-based schemes (NPNC and PNC), PICMM achieves much better performance especially when N_{data} is large because caching saves much of the uplink cost for query processing.

Overall Figs. 10–12 show that our proposed PICMM scheme performs better than NPNC, PNC and NPC, particularly as the query rate ($\lambda_{q,i}$), MH mobility rate (σ), or the cache size (N_{data}) increases. Below we reveal conditions under which PICMM could perform worse than non-caching-based schemes (NPNC and PNC).

In Fig. 13 we compare PICMM vs. NPNC, PNC and NPC management schemes in the network traffic generated, as a function of μ_i with all other parameters fixed. Under a non-caching-based scheme (NPNC or PNC), the MH will always send queries to the server for processing, since the MH does not cache data objects. As a result, the generated network traffic is insensitive to μ_i. On the other hand, when caching is used as in PICMM or NPC, the generated network traffic becomes sensitive to μ_i because the query traffic to the server depends on if cached data objects are valid. We see that PICMM consistently performs better than NPC. However, when the data update rate is very high, most cached data objects are invalid, so queries will need to be routed to the CN. In this case, there exists a cross-over point in the update rate beyond which PICMM would perform worse than a non-caching scheme because of the CN–proxy–MH triangular cost for routing query inquiries/replies and invalidation reports. In general in cases data are being updated frequently, a caching scheme would not perform better than a non-caching scheme, even with the use of a smart proxy as in PICMM for optimized, integrated cache and mobility management. In this extreme case, the system is better off with the proxy no-caching scheme (PNC).

In Fig. 14 we compare PICMM vs. NPC, PNC and NPNC in the generated network traffic, as a function of the sleep ratio ω_w/ω_s. When the sleep ratio ω_w/ω_s is extremely large, MH is mostly sleeping all the time. The cache is mostly invalid due to long sleep. In this extreme condition, PICMM incurs a higher query cost than

non-caching schemes although the cost is small since P_{wake} is small. Again we see that there is a cross-over point in the sleep ratio beyond which PICMM would perform worse than a non-caching scheme such as PNC because of the CN–proxy–MH triangular cost for routing query inquiries/replies and invalidation reports under PICMM. We see that, however, under a reasonable range of sleep ratio (<2), PICMM outperforms all other schemes.

In summary, we observe that our proposed PICMM scheme outperforms NPC, PNC and NPNC in terms of the network traffic cost generated over a wide range of parameter values, the effect of which is especially pronounced when the data update rate is low, the mobility rate is high, the query rate is high, the cache size is large, and/or the sleep ratio is low. PICMM greatly reduces the network traffic of mobile query-based applications, except under extreme conditions in which data update rates are exceptionally high and the MH disconnects more than 2/3 of the time during the execution of mobile applications. These extreme conditions, however, rarely happen in practical mobile applications.

5.3. Proxy maintenance cost

In this section we analyze the overhead generated from maintaining the per-client proxies. Let $C_{i,\text{proxyOverhead}}$ be the cost for maintaining the proxy given that the MH is in state i. Then, the average cost for maintaining the per-client proxy, denoted by $C_{\text{proxyOverhead}}$, may be calculated as a weighted average as follows:

$$C_{\text{proxyOverhead}} = \sum_i (P_i \times C_{i,\text{proxyOverhead}}). \tag{13}$$

If in state i, Mark(Xs) $<$ K, then the MH will only inform the proxy of the CoA address change, so there is no overhead. If Mark(Xs) $=$ K, then the location handoff also triggers a service handoff, the overhead is a context transfer cost of *ProxyCache* while moving the proxy to the new service area. When the MH is in the sleep state, there is no overhead. When the MH just wakes up, it will look for its proxy. If the proxy is in the same subnet as MH currently resides because the MH does not move during sleep, there is no overhead. Otherwise, the proxy is moved to the current subnet, the overhead is the context transfer cost. Therefore,

$$C_{i,\text{proxyOverhead}} = \begin{cases} 0 & \text{if Mark(Sleep)} > 0 \\ 0 & \text{else if Mark(Just Wake Up)} > 0 \text{ and Mark(Xs)} = 0 \\ F(\text{Mark}(Xs))n_{CT}\tau & \\ \quad \text{else if Mark(Just Wake Up)} > 0 \text{ and Mark(Xs)} > 0 & \\ 0 & \text{else if } Mark(Xs) < K \\ F(K)n_{CT}\tau & \text{else if } Mark(Xs) = K. \end{cases}$$

Fig. 15 shows the maintenance cost vs. the overall cost as a function of the MH's SMR denoting the relative rate at which the MH queries the database server compared with its movement. When SMR is low, it means that the MH moves across subnets frequently in between queries. Correspondingly Fig. 16 shows the percentage of proxy maintenance cost vs. the overall cost as a function of SMR. We see that the amortized cost for maintaining the per-client proxy is at its minimum when the MH's SMR is high under which the MH moves frequently in between queries, which increases the chance of proxy maintenance. For a wide range of SMR values, we observe that the proxy maintenance overhead is only a small percentage (less than 1%) of the overall cost for mobility and service management.

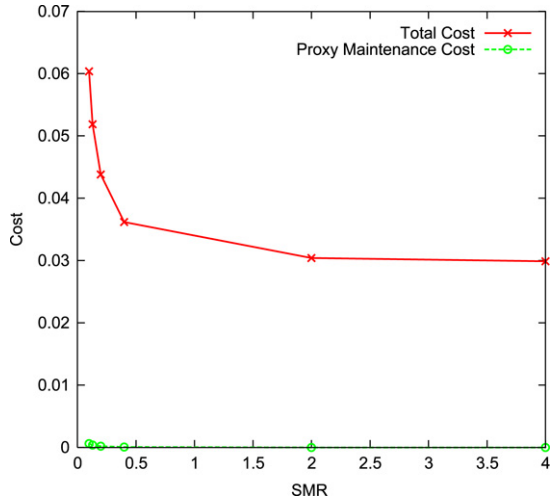

Fig. 15. Proxy maintenance cost vs. overall cost as a function of SMR.

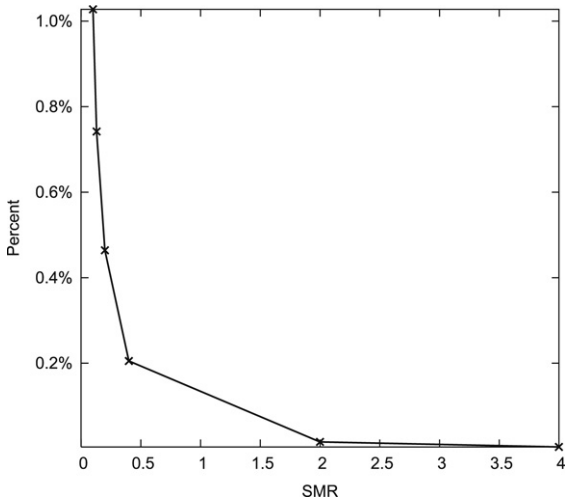

Fig. 16. Percentage of proxy maintenance cost vs. overall cost as a function of SMR.

5.4. Performance comparison: PICMM vs. decoupled schemes

To demonstrate the viability of PICMM, we compare PICMM against decoupled mobility and cache management for which mobility management is decoupled from cache management. By decoupling, we mean that the proxy for cache management is not necessarily co-located with the proxy for mobility management because the regional area size for mobility management is decoupled from that for cache management. The optimal regional size for mobility management essentially depends on the call rate to mobility rate ratio of the MH. On the other hand, the optimal regional size for service management depends on the service rate (i.e., packet rate) to mobility rate ratio of the MH. For fair comparison, we only compare them at optimal settings, i.e., for the decoupled scheme, two best regional area sizes for two separate proxies (one for mobility management and one for cache management) are used, and for PICMM, the best regional area size for the integrated proxy is used. The operational and workload conditions characterized by a set of parameter values are as listed in Table 4.

We summarize our findings in Figs. 17–20, which compare PICMM with the decoupled scheme in the generated network traffic as a function of $\lambda_{q,i}$, σ, μ_i, and sleep ratio ω_w/ω_s, respectively. We see that PICMM outperforms the decoupled scheme at optimal settings in the amount of network traffic

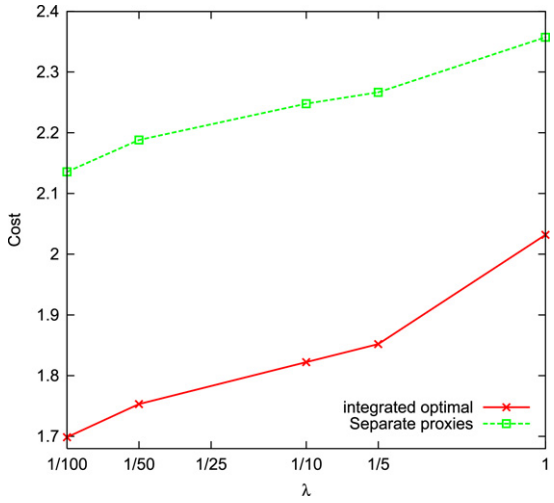

Fig. 17. PICMM vs. decoupled: Effect of $\lambda_{q,i}$.

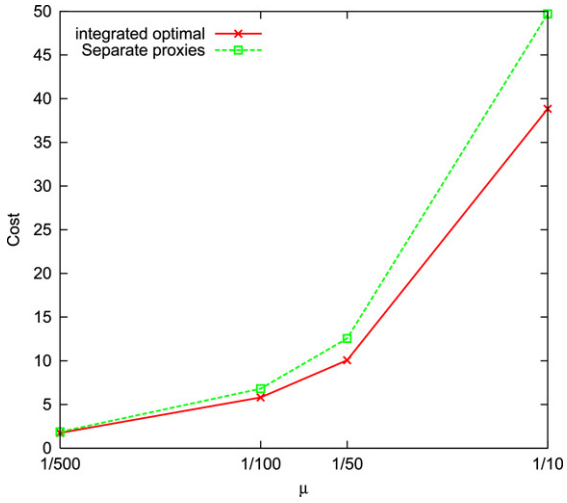

Fig. 19. PICMM vs. decoupled: Effect of μ_i.

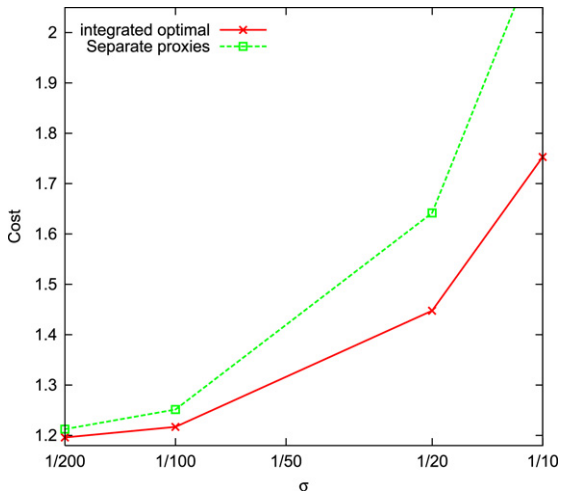

Fig. 18. PICMM vs. decoupled: Effect of σ.

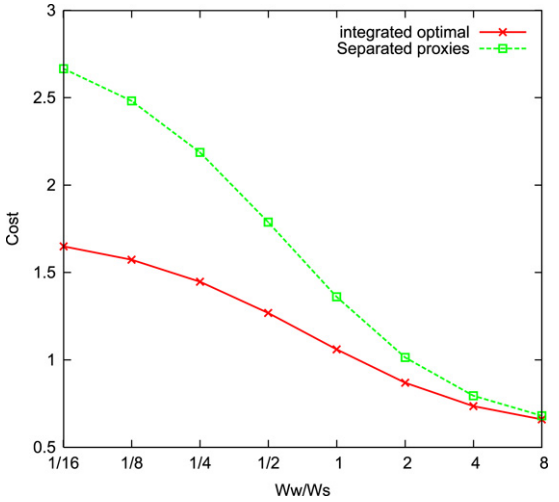

Fig. 20. PICMM vs. decoupled: Effect of ω_w/ω_s.

generated per unit time over a wide range of $\lambda_{q,i}$, σ, μ_i, and ω_w/ω_s, the effect of which is especially pronounced as when mobility rate σ and object update rate μ_i are high, and when the sleep ratio ω_w/ω_s is low. Most notably, we see that the cost difference becomes more significant as the MH mobility rate increases because of the extra overhead introduced by the decoupled system to separately track the location of the MH in cache management.

6. Applicability and conclusion

In this paper, we proposed a novel scheme for integrated mobility and service management in Mobile IPv6 systems. The core of the idea lies in leveraging a client-side proxy with duties for both cache and mobility management, and allowing intelligent MHs to determine the best service areas for service handoffs to minimize the network traffic cost. We devised a computational procedure to compute the optimal service area size that would minimize the overall network traffic cost, when given a set of parameters characterizing the MH's mobility and service characteristics. We compared our scheme with several no-proxy and/or no-caching schemes and a decoupled scheme that optimally but separately manages mobility and service activities and concluded that our scheme outperforms these schemes in terms of the network traffic cost generated over a wide range of parameter values that typify practical mobile applications.

To apply the analysis results presented in the paper, one can execute the computational procedure at static time to determine the optimal K_{opt} over a possible range of parameter values for key parameters such as α, β, n_{CT}, N_{data}, $\lambda_{q,i}$, σ and μ_i. Then at runtime an MH can perform a simple look-up operation to determine K_{opt} based on data collected at runtime. This allows each MH to dynamically determine the best service area size to minimize the overall network traffic generated. The performance gain is in the amount of network traffic communication cost saved per time unit per user, so the cost saving due to a proper selection of the best service area dynamically will have significant impacts since the cumulative effect for all mobile users over a long time period would be significant.

Our approach requires cooperation between service providers and network operators because network operators own Mobile IP network routers which do not necessarily open to service providers. The current trend, however, is that network operators also wish to provide wireless services to end users, so the service provider and network operator are often integrated as well [22]. More and more network operators also intend to offer their own proprietary services [4]. An example is modern cell phone companies that own mobile networks and also provide bundled services to cell phone users [26]. The tremendous performance gain in terms of reduced network traffic generated by our approach compared with a strict layering approach typified by the no-proxy scheme provides a strong justification of proxy-based

integrated cache consistency and mobility service management for client–server applications in Mobile IP networks.

In the future, we plan to investigate fault tolerance aspects of integrated service and mobility management for caching-based mobile applications. We also plan to investigate security issues including establishing and maintaining trust between the client and the proxy and preventing man-in-the-middle attacks.

References

[1] I. Akyildiz, J. Xie, S. Mohanty, A survey on mobility management in next generation all-IP based wireless systems, IEEE Wireless Communications 11 (4) (2004) 16–28.

[2] R.J. Bates, Broadband Telecommunications Handbook, McGraw-Hill Professional, ISBN: 0071398511, 2002.

[3] E. Borgia, M. Conti, F. Delmastro, Mobileman: Design, integration, and experimentation of cross-layer mobile multihop ad hoc networks, IEEE Communications 44 (7) (2006) 80–85.

[4] K. Buchanan, R. Fudge, D. Mcfarlane, T. Phillips, A. Sasaki, H. Xia, IMT-2000: Service provider's perspective, IEEE Wireless Communications 4 (4) (1997) 8–13.

[5] G. Cao, A scalable low-latency cache invalidation strategy for mobile environments, IEEE Transactions on Knowledge and Data Engineering 15 (5) (2003) 1251–1265.

[6] I.-R. Chen, W. He, B. Gu, Proxy-based regional registration for integrated mobility and service management in Mobile IP systems, The Computer Journal 50 (3) (2007) 281–293.

[7] H. de Meer, A. Corte, A. Puliafito, O. Tomarchio, Programmable agents for flexible QoS management in IP networks, IEEE Transactions on Selected Areas in Communications 18 (2) (2000) 256–267.

[8] B. Gu, I.R. Chen, Performance analysis of location-aware mobile service proxies for reducing network cost in personal communication systems, ACM Mobile Networks and Applications 10 (4) (2005) 453–463.

[9] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, B. Patil, Proxy mobile IPv6, IETF (2008) (in preparation) http://www.ietf.org/rfc/rfc5213.txt.

[10] E. Gustafsson, A. Jonsson, C. Perkins, Mobile IPv4 regional registration, IETF (2006) (in preparation).

[11] W. He, I. Chen, B. Gu, A proxy-based integrated cache consistency and mobility management scheme for Mobile IP systems, in: The IEEE 21st International Conference on Advanced Information Networking and Applications, Niagara Falls, Canada, 2007, pp. 354–361.

[12] D.M. Huizinga, P. Mann, Disconnected operation for heterogeneous servers, in: ACM symposium on Applied Computing, 1996, pp. 312–321.

[13] J. Jing, A. Elmagarmid, A. Helal, R. Alonso, Bit-sequences: An adaptive cache invalidation method in mobile client/server environments, ACM Mobile Networks and Applications (1997) 115–127.

[14] D. Johnson, C. Perkins, J. Arkko, Mobility Support in IPv6, IETF (2004) (in preparation) http://www.ietf.org/rfc/rfc3775.txt.

[15] K. Kong, W. Lee, Y. Han, M. Shin, H. You, Mobility management for all-IP mobile networks: Mobile IPv6 vs. Proxy Mobile IPv6, IEEE Wireless Communications 15 (2) (2008) 36–45.

[16] A. Misra, S. Das, A. McAuley, A. Dutta, S.K. Das, IDMP: An intra-domain mobility management protocol using mobility agents, draft-mobileipmisra-idmp-00.txt, IETF (2000) (in preparation).

[17] S. Mohanty, I. Akyildiz, Performance analysis of handoff techniques based on Mobile IP, TCP-Migrate and SIP, IEEE Transactions on Mobile Computing 6 (7) (2007) 731–747.

[18] S. Mtika, F. Takawira, Mobile IPv6 regional mobility management, in: ACM 4th international symposium on Information and communication technologies, Cape Town, South Africa, 2005, pp. 93–98.

[19] X. Perez-Costa, M. Torrent-Moreno, H. Hartenstein, A performance comparison of Mobile IPv6 Hierarchical Mobile IPv6, fast handovers for Mobile IPv6 and their combination, SIGMOBILE Mobile Computing Communications Review 7 (4) (2003) 5–19.

[20] E. Pitoura, G. Samaras, Data Management for Mobile Computing, Kluwer Academic Publishers, 1998.

[21] R. Ramjee, K. Varadhan, L. Salgarelli, S. Thuel, S. Wang, T.L. Porta, HAWAII: A domain-based approach for supporting mobility in wide-area wireless networks, IEEE/ACM Transactions on Networking 10 (3) (2002) 396–410.

[22] H.K. Sabat, The evolving mobile wireless value chain and market structure, Telecommunications Policy 26 (9–10) (2002) 505–535.

[23] H. Soliman, C. Castelluccia, K. El-Malki, L. Bellier, Hierarchical mobile IPv6 mobility management, IETF (in preparation) (2005) http://www.ietf.org/rfc/rfc4140.txt.

[24] K. Tan, J. Cai, B. Ooi, An evaluation of cache invalidation strategies in wireless environments, IEEE Transactions on Parallel and Distributed Systems 12 (8) (2001) 789–807.

[25] A. Valko, Cellular IP: A new approach to Internet host mobility, ACM Mobile Computing Communications Review 29 (1) (1999) 50–65.

[26] U. Varshney, R. Vetter, Mobile commerce: Framework applications and networking support, Mobile Networks and Applications 7 (3) (2002) 185–198.

[27] Z. Wang, M. Kumar, S. Das, H. Shen, Dynamic cache consistency schemes for wireless cellular networks, IEEE Transactions on Wireless Communications 5 (2) (2006) 366–376.

[28] Z. Wang, K. Lan, R. Berriman, T. Moors, M. Hassan, L. Libman, M. Ott, B. Landfeldt, Z. Zaidi, A. Seneviratne, Experiences in deploying a wireless mesh network testbed for traffic control, ACM SIGCOMM Computer Communication Review 37 (5) (2007) 17–28.

[29] J. Xie, I. Akyildiz, A novel distributed dynamic location management scheme for minimizing signaling costs in Mobile IP, IEEE Transactions on Mobile Computing 1 (3) (2002) 163–175.

[30] J. Xu, X. Tang, D. Lee, Performance analysis of location-dependent cache invalidation schemes for mobile environments, IEEE Transactions on Knowledge and Data Engineering 15 (2) (2003) 474–488.

[31] M. Yeung, Y. Kwok, Wireless cache invalidation schemes with link adaptation and downlink traffic, IEEE Transactions on Mobile Computing 4 (1) (2005) 68–83.

[32] P. Zerfos, G. Zhong, J. Cheng, H. Luo, S. Lu, J.J.-R. Li, Dirac: A software-based wireless router system, in: 9th annual international conference on Mobile computing and networking, 2003, pp. 230–244.

[33] X. Zhang, J. Castellanos, A. Campbell, P-MIP: Paging extensions for Mobile IP, Mobile Networks and Applications 7 (2) (2002) 127–141.

Weiping He received his B.E. degree in Electrical Engineering from Huazhong University of Science and Technology, China, in 1996, and his M.S. and Ph.D. degrees in Computer Science from Virginia Tech in 2002 and 2009, respectively. His research interests include mobile computing, wireless networks, performance analysis, mobile data management, and mobility and service management in mobile computing environments.

Ing-Ray Chen received his B.S. degree from the National Taiwan University, Taipei, Taiwan, and his M.S. and Ph.D. degrees in computer science from the University of Houston. He is a professor in the Department of Computer Science at Virginia Tech. His research interests include mobile computing, security, multimedia, wireless networks, real-time intelligent systems, and reliability and performance analysis. Dr. Chen has served as program chair and program committee member for numerous international conferences. Dr. Chen currently serves as an editor for Wireless Personal Communications, The Computer Journal, Wireless Communications and Mobile Computing, International Journal of Security and Network Communications, and International Journal on Artificial Intelligence Tools. He is a member of the IEEE/CS and ACM.