Mobile Networks and Applications 9, 627–631, 2004

Hash-Based Paging and Location Update Using Bloom Filters

A paging algorithm that is best suitable for IPv6

PARS MUTAF and CLAUDE CASTELLUCCIA
INRIA Rhône-Alpes, Planète Team, 655 avenue de l'Europe, Montbonnot, 38334 Saint Ismier Cedex, France

Abstract. We develop and analyze a hash-based paging and location update technique that reduces the paging cost in cellular systems. By applying a Bloom filter, the terminal identifier field of a paging message is coded to page a number of terminals concurrently. A small number of terminals may wake up and send what we call "false location updates" although they are not being paged. We compare the total number of paging and false location update messages with the cost of the standard paging procedure. Fortunately, the false location update probabilities can be made very small, and important bandwidth gains can be expected. The larger the size of the terminal identifier, the less probable are false location updates. Therefore, hash-based paging especially shows promise for IP paging in mobile IPv6 networks with 128-bit mobile host addresses.

Keywords: bloom filters, concurrent paging, false location update, IP paging, MIPv6

1. Introduction

In cellular systems, terminals must update the network with their current location in order to receive packets. Idle mobile terminals update the network only when they cross paging area boundaries, which cover a relatively large number of cells. Since terminals are idle most of the time, paging reduces the bandwidth cost of location updates and battery drain on mobile terminals. The cost being paid for this service is the bandwidth cost of broadcasting a paging message in each cell of a paging area upon call arrival. A paged terminal reports its exact location by sending a location update message and the communication begins. This procedure is illustrated in figure 1. Upon call arrival, a terminal *id* is paged in its paging area that covers the base stations $BS_1, BS_2, \ldots, BS_w$ by broadcasting a paging message denoted *Pmes*. The callee detects its identifier (*id*) in the paging message and reports its exact location (*loc*) using a location update message denoted *LUmes*.

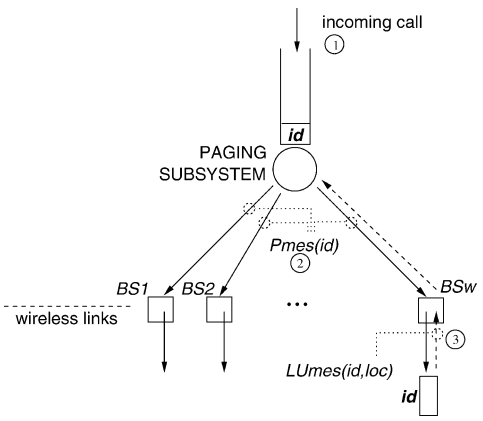


Figure 1. Standard paging and location update procedure.

A better paging service demands larger paging areas. However, large paging areas create a permanently high incoming call rate per paging area (hence, high paging load) since a large amount of terminals are served per paging area. The paging load may also temporarily increase due to flash crowds. Coping with high paging load requires reserving an important portion of the available bandwidth in each cell for paging messages. Otherwise, paging processes are queued in the network and important call setup delays are incurred, which is also undesirable. This problem has received considerable attention and important research has been done for reducing the bandwidth cost of paging messages.

In this paper, we propose a hash-based paging and location update procedure (or, hash-based paging) using Bloom filters [1]. A Bloom filter is a randomized data structure for concisely representing a set, in order to support membership queries. The space efficiency is achieved at the cost of a small probability of false positive. We note that Bloom filters can be used to page several terminals in a same paging area concurrently, without modifying the original paging message size. Paging bandwidth consumption and the call setup delays can be significantly reduced.

The remaining sections of this paper are organized as follows: section 2 describes hash-based paging, section 3 analyzes the bandwidth gain that can be obtained, section 4 presents a discussion on IP Paging and Mobile IPv6 with large identifier sizes, and finally section 6 presents some conclusions.

2. Hash-based paging and location update

We propose using Bloom filters [1] for concurrently paging a set $A = \{id_1, id_2, \ldots, id_n\}$ of n terminals that reside in a same paging area. The m-bit terminal identifier field *ID* found

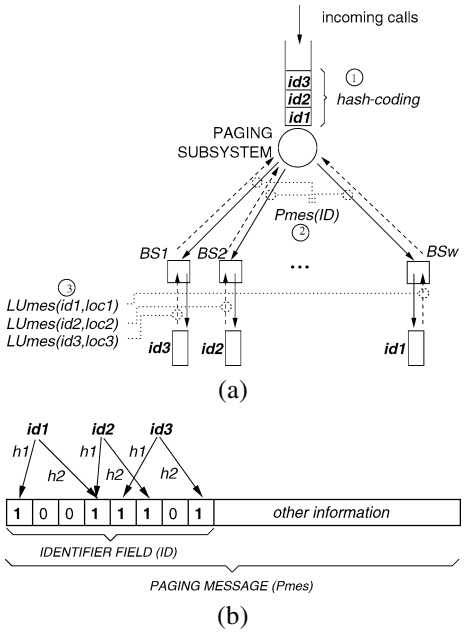


Figure 2. Hash-based paging and location update using Bloom filters: (a) concurrent paging and location update; (b) hash-coding (with $k = 2$).

in a paging message *Pmes* is coded to represent all members of the set A. The procedure requires k independent uniform hash functions, $h_1(), h_2(), \ldots, h_k()$ (where, $1 \leqslant h() \leqslant m$). First, all bits of *ID* are set to 0, then for each element $id_i \in A$, the bits at positions $h_1(id_i), h_2(id_i), \ldots, h_k(id_i)$ in *ID* are set to 1 (a particular bit may be set multiple times). The resulting vector *ID* can be used to page all members of A concurrently by broadcasting a standard paging message. Upon receipt of the broadcast paging message, a given terminal id_j can detect if it is being paged by checking the bit positions $h_1(id_j), h_2(id_j), \ldots, h_k(id_j)$ in *ID*. If any of them is 0, then id_j is certainly not being paged. Otherwise, id_j is being paged and should respond with a location update message, *LUmes*. The hash-based paging and location update procedure of $n = 3$ terminals is illustrated in figure 2.

The space-efficiency of Bloom filters is achieved at the cost of a small *false positive* probability. I.e. there is a small probability that a terminal id_j receives a paging message with set *ID* bits at positions $h_1(id_j), h_2(id_j), \ldots, h_k(id_j)$ although $id_j \notin A$. A false positive leads to what we call a *false location update*, since the location update response of id_j is useless (id_j is not paged). The false positive (or, false location update in our case) probability F and the optimal number of hash functions k_{opt} that minimizes F are well-known [3]

$$F = \left(1 - \mathrm{e}^{-kn/m}\right)^k \tag{1}$$

and minimized for

$$k_{\text{opt}} = (\ln 2) \cdot \frac{m}{n} \tag{2}$$

in which case it becomes

$$F = (0.6185)^{m/n}. \tag{3}$$

Since a false location update will result in unnecessary bandwidth and energy consumption, k should be chosen to minimize F regarding the number of concurrently paged terminals (by equation (2)). In practice, this can be easily achieved by fixing the maximum number of hash functions $k_{\max}$ and transmitting k in paging messages. The number of hash computations will have no impact on terminal energy consumption. A terminal id_j can store the results of $h_1(id_j), h_2(id_j), \ldots, h_{k_{\max}}(id_j)$ and use the first k results for future paging events. On the network side, the computational load due to hash computations will be self-tuning since the optimal number of hash functions decreases with increasing paging load (i.e. n).

According to equation (3), the false location update probability will still increase with the number of concurrently paged terminals, however good performance can be expected if the terminal identifier size is large enough.

3. Bandwidth gain

Using the standard paging procedure, the paging cost in each cell of a paging area is one paging message per incoming call. In hash-based paging, the paging cost in each cell is one paging message and several false location update messages per n incoming calls. Let d the terminal density, i.e. the number of terminals per cell, then the paging cost in each cell is $1 + d \cdot F$ per n incoming calls. The bandwidth gain (simplified for equal sized paging and location update messages) is:

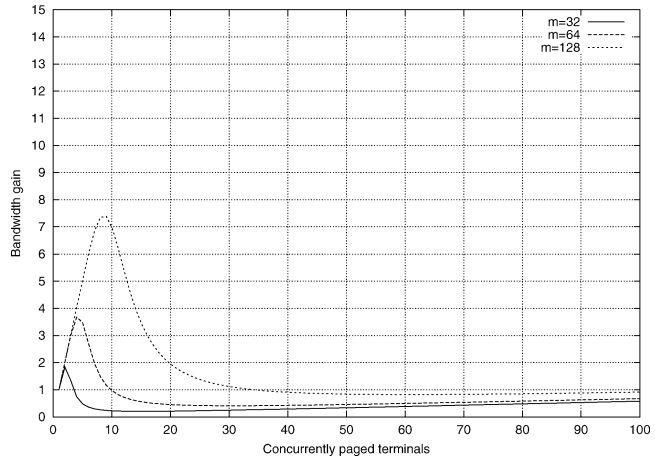
$$G = \frac{n}{1 + d \cdot F}. \tag{4}$$

Figure 3 shows the bandwidth gain for different identifier sizes and terminal densities. It is assumed that in macro and micro cellular environments, the terminal densities are $d = 200$ and $d = 20$, respectively [9]. Note that hash-based paging will be more attractive as cell sizes get smaller, which is justified by the current trend in support of increasingly important number of cellular users.

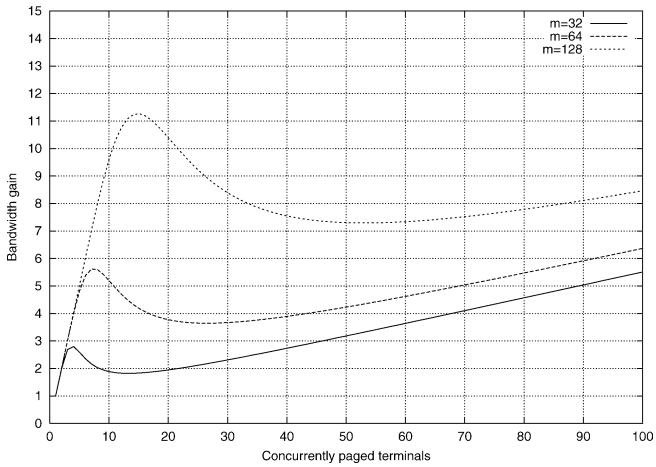
In practice, the number of concurrently paged terminals should satisfy the constraint $F \simeq 0$ so that false location updates introduce a negligible energy consumption overhead. This roughly corresponds to the interval where $\frac{\mathrm{d}G}{\mathrm{d}n} \simeq 1$ in figure 3. We are satisfied with $F \leqslant 0.001$, since a given terminal will not be unnecessarily awakened more than once per 1000 paging events. By equation (3) it is easily shown that, setting

$$n \leqslant 0.06955m \tag{5}$$

will meet our constraint. Tables 1–3 show the bandwidth gain that can be obtained under energy constraints and for integer k values. Important bandwidth can be saved with negligible energy consumption overhead. Each table shows the bandwidth gains for n values that introduce tolerable false location update probabilities i.e. $F \leqslant 0.001$. For example, with $m = 32$, setting $n = 3$ leads to $F = 0.005947$. Hence it is not recommended in table 1. Note however that in table 3, we recommend $n = 9$ setting since the false location update probability is very close to 0.001.



(a)



(b)

Figure 3. Bandwidth gain: (a) macro cells; (b) micro cells.

Table 1
$m = 32$.

			G	
n	k_{opt}	F	$d = 20$	$d = 200$
2	11	0.000459	1.981821	1.831959

Table 2
$m = 64$.

			G	
n	k_{opt}	F	$d = 20$	$d = 200$
2	22	0.000000	1.999992	1.999916
3	14	0.000035	2.997879	2.978927
4	11	0.000459	3.963642	3.663917

4. IP paging and MIPv6

The proposed optimization may be applicable to different access technologies, therefore we do not loose generality in this paper. The potential gain of hash-based paging is considerable even when identifier sizes are relatively small, e.g., $m = 32$, or $m = 64$. For example, table 1 promises a paging bandwidth gain of $G \simeq 2$ in each cell. While we promote hash-based paging for small identifier sizes as well, this optimization is most attractive for IP paging in MIPv6 (mobile IPv6) networks with 128-bit mobile host addresses.

Table 3
$m = 128$.

			G	
n	k_{opt}	F	$d = 20$	$d = 200$
2	44	0.000000	2.000000	2.000000
3	29	0.000000	3.000000	2.999999
4	22	0.000000	3.999983	3.999832
5	17	0.000005	4.999545	4.995451
6	14	0.000035	5.995759	5.957855
7	12	0.000153	6.978653	6.792231
8	11	0.000459	7.927285	7.327835
9	9	0.001078	8.810141	7.404357

IP paging [5,8,9] is an ongoing research topic which has been proposed for MIPv4 [6] and MIPv6 [4] networks that support multiple access technologies and IP-based micro-mobility protocols [2,7]. MIPv4 and MIPv6 define a *care-of-address* that identifies a mobile host in its current subnet and a global identifier called *home address*. In IP paging, a paging message *Pmes* carries the IP address of the callee (home or care-of address depending on the paging protocol design) and processed by the IP modules of dormant hosts. If the IP address matches, the destination host detects that it is being paged and responds. Hash-based paging can improve the IP paging performance. For example, in P-MIP [9], MIPv4 hosts are paged using their 32-bit home addresses. The performance of P-MIP can be increased by a factor of $G \simeq 2$ at little implementation cost. However, this optimization is especially attractive in the case of MIPv6 which defines 128-bit home and care-of addresses. Table 3 indicates that a bandwidth gain of $G = 8.81$ or $G = 7.40$ can be achieved at negligible false location update overhead.

5. Simulation analysis of paging delays

In this section we assume that paging messages are rate limited, in which case the bandwidth cost of broadcast paging messages is controlled at the risk of paging delays. We simulate one hour of incoming sessions in a paging area and observe the paging delays with three different paging area sizes, with and without Bloom filter optimization.

The paging rate in a paging area is limited to $R = 1$ paging/second. As a consequence, an incoming call rate higher than R results in queueing delays. As we increase the paging area size, we expect larger incoming call rates (hence, call setup delays). When hash-based paging is used, a paging message pages all terminals found in the paging queue. I.e. the paging queue is emptied every second by broadcasting a single paging message. Thus, the upper bound of paging delay is 1 second. The cost to pay is hash computation cost (by the network) and false location update risk.

Figure 4 shows the observed frequencies of different paging delays (in log scale) with paging areas that cover different number of terminals N. Incoming calls of each terminal are

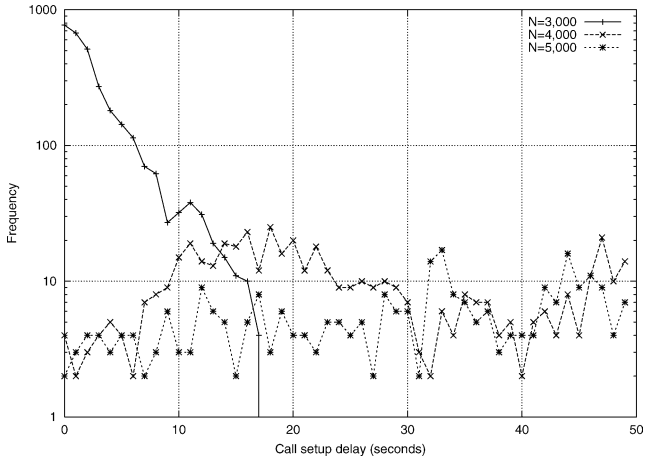


Figure 4. Paging delays with standard paging.

Table 4
Hash computation cost.

N	*hps*
3,000	50.03
4,000	59.27
5,000	66.36

driven by a Poisson process and the average incoming call rate is set to 1 call/hour. The terminal identifier size is set to $m = 128$. With a paging area size that covers $N = 3{,}000$ terminals, paging delays up to 17 seconds were observed. 4 calls suffered from 17 seconds of paging delay. A total of 160 calls suffered from more than 10 seconds delay. With larger paging area sizes, standard paging was unsuccessful. The number of calls in the paging queue constantly increased, and excessive paging delays were incurred (paging delays more than 50 seconds are not shown).

The hash computation rates (hash per second, denoted *hps*) with hash-based paging are shown in table 4. We have not observed any false location update during these simulations.

Figure 5 shows the number of calls waiting in the paging sub-system for $N = 3{,}000$. Using hash-based paging, the number of queued calls never exceeded $n = 5$. Thus, the false location update risk was $F \leqslant 0.000005$ (according to table 3). With $N = 4{,}000$ and $N = 5{,}000$, the number of queued calls were mostly less than or equal to 5. Larger values were 6 and 7 (occurred twice and once) and 7 (occurred 3 times), respectively. The false location update probabilities with $n = 6$ and $n = 7$ were $F = 0.000035$ and $F = 0.000153$, respectively, and hence did not occur during one hour simulation.

6. Conclusion

In this paper, we have developed and analyzed a hash-based paging and location update procedure using Bloom filters. Multiple terminal identities are coded into a single terminal identifier found in a standard paging message and paged concurrently. Hash-based paging reduces the rate of broadcast paging messages and queuing delays in the network, at

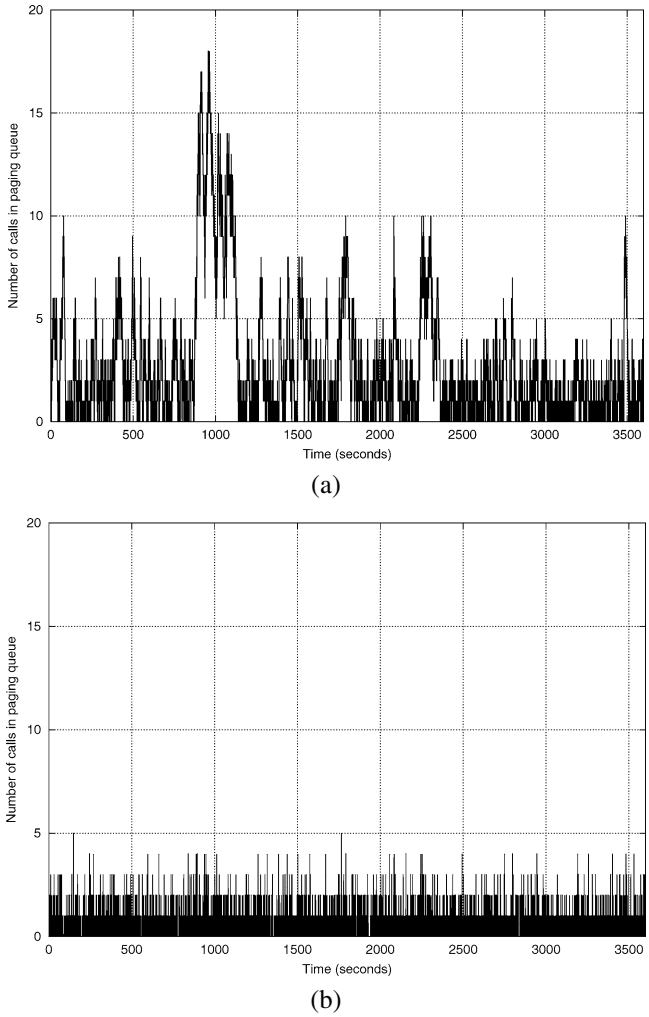


Figure 5. Number of calls in paging queue: (a) standard paging; (b) hash-based paging.

the cost of introducing some unnecessary i.e. "false" location update messages. Despite false location updates, important bandwidth gains can be obtained with negligible energy consumption overhead. The proposed optimization is especially attractive for IP paging in Mobile IPv6 networks with 128-bit mobile host addresses. However, hash-based paging also shows promise for smaller identifier sizes, e.g., 32 or 64-bit terminal identifiers.

References

[1] B. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13(7) (July 1970) 422–426.

[2] A.T. Campbell et al., Design, implementation, and evaluation of cellular IP, IEEE Personal Communications, Special issue on IP-based mobile telecommunications networks (June/July 2000).

[3] L. Fan, P. Cao, J. Aldeida and A. Broder, Summary cache: A scalable wide-area web cache sharing protocol, in: *Proceedings of SIGCOMM'98 Conference*, Vol. 28 (1998) pp. 254–265. Corrected version available at URL: http://www.cs.wisc.edu/~cao/papers/summarycache.html

[4] D. Johnson, C. Perkins and J. Arkko, Mobility support in IPv6, Internet draft, draft-ietf-mobileip-ipv6-19.txt (work in progress, October 2002).

[5] J. Kempf, Dormant mode host alerting (IP paging) problem statement, RFC 3132 (June 2001).
[6] C. Perkins, IP mobility support, RFC 3344 (August 2002).
[7] R. Ramjee et al., HAWAII: A domain-based approach for supporting mobility in wide-area wireless networks, IEEE/ACM Transactions on Networking 6(2) (June 2002).
[8] R. Ramjee et al., IP paging service for mobile hosts, in: *Proceedings of MOBICOM'2001*, Rome, Italy (July 2001).
[9] X. Zhang, J. Castellanos and A. Campbell, P-MIP: Paging extensions for mobile IP, ACM Mobile Networks and Applications (MONET) 7(2) (March 2002).

Claude Castelluccia
E-mail: claude.castelluccia@inria.fr

Pars Mutaf
E-mail: pars.mutaf@inria.fr