Optimisation de la signalisation dans la gestion de la mobilité IPv6

Pars Mutaf

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque
/_/_/_/_/_/_/_/_/_/

THESE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité: Informatique, Systèmes et Communications

Préparée à l'INRIA Rhône Alpes (Projet Planète), dans le cadre de l'école doctorale "Mathématiques, Sciences et technologies de l'information, Informatique"

Présentée et soutenue publiquement par

PARS MUTAF

le 15 Janvier 2004

Titre:

Optimisation de la Signalisation dans la Gestion de la Mobilité IPv6

(Signaling Optimizations for Neighbor Discovery, Paging and Mobility Management in IPv6)

Directeur de thèse:
M. Claude Castelluccia

Jury:

M. Andrzej Duda	*Président*
M. Torsten Braun	*Rapporteur*
M. Hossam Afifi	*Rapporteur*
M. Gabriel Montenegro	*Examinateur*

Abstract

In this thesis we address some potential signaling optimization problems in a wireless Internet based on IPv6 and Mobile IPv6. Mobile hosts need to update the network with their location (subnet) in order to receive packets. Therefore, with the introduction of millions of highly mobile IP devices to the Internet, a high rate of location update signaling is expected in the core network. This problem can be addressed by configuring large subnets (or paging areas) that cover many wireless access points. In this case however, excessive address resolution and IP paging signaling may consume the more limited bandwidth in the wireless access network. Both address resolution and IP paging consist of broadcasting a query over all cells of a subnet in order to reestablish connectivity with a mobile host. Thus, in a large subnet with many mobile hosts, we expect a high broadcast rate although each mobile host rarely receives incoming sessions. We address this problem using a technique that we call "hash-based broadcast". In hash-based broadcast, we represent multiple IPv6 addresses using a single 128-bit Bloom filter, which allows the network to broadcast a same query to multiple IPv6 hosts without increasing the original message size. This technique is best suitable for IPv6 since Bloom filter performance increases with its size.

We also develop an adaptive IP paging architecture for dynamically configuring network-aggregated and variable sized paging areas. Variable sized paging areas can be used for adaptive IP paging, in which paging area sizes are adapted to each host's mobility and incoming session rates.

We note that end-to-end IP paging is possible, and a good candidate for countering home agent failure in Mobile IPv6. In Mobile IPv6, a mobile host becomes unreachable without home agent service, which reduces the robustness of the protocol. By replacing the paging area concept, by a living area, we define an end-to-end IP paging protocol. In case of home agent failure a correspondent node can page a mobile host in the subnets that form its living area. Most mobile hosts have a more or less fixed living area that they rarely leave. As a result, end-to-end IP paging is useful because it is likely to succeed in most of the cases.

Finally, we propose a transparent denial-of-service protection for Mobile IPv6 hosts and networks. We adapt the SYN-cookies approach, that is used to protect public Internet servers, to Mobile IPv6. A home agent acts as a proxy on the first packet of an incoming session by returning a SYN-cookie. Unless the correspondent node acknowledges the right cookie, the session initiating packet is not routed to the destination mobile host. This scheme is useful, because in Mobile IPv6 a mobile host is not easily reachable to attackers, without home agent service. The home agent, in this case, acts as single point of entry where incoming sessions can be secured against denial-of-service attacks.

ACK	Acknowledgment
AH	Authentication Header
ARP	Address Resolution Protocol
CCN	Critical Correspondent Node
CDMA	Code Division Multiple Access
CGA	Cryptographically Generated Address
COT	Care-of Test
COTI	Care-of Test Init
CPU	Central Processing Unit
DAD	Duplicate Address Detection
DOS	Denial-of-Service
E2EP	End-to-End IP Paging
ESP	Encapsulating Security Payload
FMIPv6	Fast Handovers for Mobile IPv6
FTP	File Transfer Protocol
GPRS	General Packet Radio Service
GSM	Global System for Mobile telecommunications.
HAF	Home Agent Failure
HBAR	Hash-Based Address Resolution
HMIPv6	Hierarchical Mobile IPv6
HOT	Home-of Test
HOTI	Home-of Test Init
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ID	Interface Identifier
IKE	Internet Key Exchange
IMAP	Internet Message Access Protocol
IP	Internet Protocol (version 4 or 6)
IPsec	IP security
IPv6	Internet Protocol version 6
ISN	Initial Sequence Number
LA	Living Area
LPA	Largest Paging Area
MAP	Mobility Anchor Point
MH	Mobile Host
MIP	Mobile IP
MIPv6	Mobile IPv6
MSS	Maximum Segment Size
PACA	Paging Area Configuration Agent
PKI	Public Key Infrastructure
POP	Post Office Protocol
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SYN	Synchronization
TCM	Threshold Controlled Mode
TCP	Transmission Control Protocol
UBM	Upper Bounded Mode
UDP	User Datagram Protocol
UHA	Unavailable Home Agent
VoIP	Voice over IP
VPN	Virtual Private Network
WSF	Window Scale Factor
WWW	World Wide Web

Contents

Chapter 1

Introduction

1.1 Mobile, wireless and IPv6-based Internet access

The popularity of the WWW and e-mail services led to the unexpected explosive growth of the Internet during the last two decades. The success of the Internet is due to the wide application space that can be built on top of the simple and access technology-independent IP (Internet Protocol) infrastructure. While the basic principles of Internet communication have not changed by much until now, a radical change in the end-host behavior is expected in a near future. Unsatisfied with voice-oriented networks such as GSM, billions of cellular phone users are expected to become Internet hosts in order to profit from the large application set provided by the Internet. These new users will expect VoIP services but also WWW access, or e-mail services (to name a few) using their hand-held devices and also constant reachability regardless of their physical location. This model makes sense for cellular operators as well, which can then profit from the end-to-end nature of the Internet protocols rather than building application-specific networks which are considered complex, unreliable and expensive. Each cellular operator can build a wireless extension to the Internet, providing a simple and well-defined IP routing service its subscribers. By pushing the complexity to the end-node i.e. host implementations, the operators can then avoid a new network extension or modification each time a new service is introduced. Mobile Internet users as well, just like any other Internet user, can obtain the necessary software to profit from the new service without changing their IP-enabled equipment.

This trend points to IP (or data) centric communication, and eventually a single communication infrastructure for mobile and non-mobile hosts, which is the Internet. However, for truly end-to-end Internet communication, every Internet host must be assigned a unique address called an IP address, at least while communicating with another host. Mobile hosts are not an exception. The current version of the Internet adopts the IPv4 protocol which allows in theory, up to 2^{32} IP addresses. This address space is considered insufficient today. IPv4 addresses are not distributed as a function of human population, and hence most of the planet is already out of IP address space. The new Internet protocol, IPv6, is being designed by the IETF (Internet Engineering Task Force) in order to overcome the address space limitations of IPv4. The IPv6 protocol allows, in theory, up to 2^{128} hosts to be constantly connected to the Internet. The address space provided by IPv6 is considered enough, hopefully until the extinction of the human race. As a consequence, in this thesis we focus our work on IPv6, since IPv6 is far more likely to satisfy the IP address space constraints in the presence of billions of IP-addressable mobile hosts.

In this thesis, we focus our attention on the IP mobility model adopted by the Mobile IPv6, the current IETF standard. Mobile IPv6 allows any Internet host to change its location i.e. subnetwork, or subnet, without loosing its reachability to its potential correspondent nodes and to maintain active sessions while moving. The Mobile IPv6 protocol does so by assigning each mobile host a globally unique and fixed home address that does not change upon movement.

Since the mobility service is provided at the IP-layer, a mobile host can use any IP-compatible link layer wireless access technology, provided that it is equipped with the required network interface card.

1.2 Problem statement

Session delivery signaling cost in the access network

It is expected that Mobile IPv6 hosts will receive any kind of sessions including real-time sessions e.g. VoIP from other Internet hosts located anywhere in the Internet. Session delivery to mobile hosts may introduce important signaling overhead in the wireless access network, depending on the session delivery technique used. In this thesis we consider two techniques which are *Address Resolution*, and *IP Paging*. Below, we overview these techniques and their signaling costs:

1. Address resolution: The IPv6 Neighbor Discovery protocol allows an access router to dynamically resolve the link layer address of a host in its subnet upon incoming session. Upon receipt of a session initiating packet from a correspondent node, the intervening access router buffers the packet and broadcasts (or multicasts) an address resolution request in its subnet. The destination host that detects the address resolution request, responds by sending its link layer address, and the access router routes the session initiating packet. At this point the link layer address of the host is cached by the access router and refreshed upon every packet routed to/from the host. As a consequence, address resolution signaling cost and delay for that host is avoided for the subsequent packets received during the session.

   In some wireless extensions to the Internet, a subnet may cover a large number of link layer wireless access points, i.e. cells and thereby cover a wide geographical area. In those cases, a very large number of mobile hosts per IP subnet can be expected. The address resolution signaling cost in a very large subnet will be two-fold: First, each incoming session to an unknown IP address (i.e. an address for which the corresponding link layer address is not available in the access router's cache) will trigger a broadcast address resolution request, hence consume bandwidth in all cells of the large IP subnet. Secondly, and more importantly, a large subnet will cover a large number of mobile hosts, each expecting incoming session from their respective correspondent nodes. Although we estimate a relatively low incoming session rate for each individual mobile host, the total incoming session rate in a subnet will grow with the number of served mobile hosts. In a very large subnet, an important portion of the available bandwidth, in each cell, may be consumed for address resolution requests. This problem is considered as a limitation on the subnet size and/or the number of users that can be served per subnet.

2. IP paging: Paging is used in current cellular systems in order to reduce the location update rate and power drain of mobile terminals. Recent work on IP mobility management considered adapting a paging mechanism to the wireless Internet using a protocol called "IP paging". IP paging is very similar address resolution. However, IP paging is more expensive in terms of broadcast signaling rate, but more optimal in terms of energy.

   In IP paging, cells are grouped into "paging areas". Subnet and paging area boundaries may or may not be identical. Within paging area boundaries, an idle mobile host does not report its current cell to the network and also enters a power-efficient "dormant" mode in which its network interface consumes much less energy. Since mobile hosts are idle most of the time, and paging areas have wide geographical coverage, IP paging greatly reduces the power consumed by the network interface.

   The cost to pay is the broadcast "paging cost" upon incoming session to an idle host. The intervening "paging agent" buffers the session initiating packet and pages the destination host by broadcasting a paging request message over the mobile host's current paging area. The

paged host detects the request, reports its exact location and the paging agent routes the buffered packet. At this point the mobile host enters active mode in which its exact location is always known to the system, and hence paging is not necessary during a session. The broadcast signaling cost of paging is similar to that of address resolution. I.e., upon every incoming session to a mobile host bandwidth is consumed in many cells, and the paging rate in a paging area grows with the number of users per paging area, hence paging area size. The difference is that, in IP paging, caching is not possible. An idle mobile host may change its location between two consecutive sessions, but it will not associate with its current access point. As a consequence, broadcast paging signaling cost will be paid upon every incoming session destined for an idle host.

Session delivery robustness problems

Mobile IPv6 introduces a new architectural element to the Internet, which is called a home agent. While away from their home subnet, mobile hosts report their current subnet to their home agent by sending a binding update message. A correspondent node that wishes to communicate with a mobile host, sends a packet to its home address, and the packet is intercepted by the mobile hosts home agent. Since the home agent is always kept up to date about the mobile hostŠs current subnet, the home agent can deliver the session.

However, the home agent, or home network is a single point of failure in this case. If the home agent is temporarily down, the correspondent node's session cannot be delivered to the mobile host. This is unfortunate because in most cases, there may be a different routing path between the correspondent node and the mobile host, i.e. an alternative and possibly shorter path that functions correctly. This problem reduces the robustness of the Mobile IPv6 protocol. However, it should be viewed as a signaling optimization problem. The question that needs answering is: How a correspondent node can itself locate i.e. page a mobile host with a minimum signaling cost?

Session delivery security problems

While optimization is important, we do not feel comfortable unless it is accompanied by some protection. A wireless Internet is, in some sense, a traditional cellular network which opens its doors to incoming sessions coming from the Internet. This is risky, since the doors are opened to many malicious and powerful Internet nodes as well.

Traditionally, optimization has been a major issue in wireless networks, since bandwidth and energy are limited resources. We expect that these resources will be easy targets. With very little effort, malicious Internet nodes can consume precious bandwidth and energy in a target wireless network.

1.3 Overview of proposals

As described above, our work in this thesis is focused on different signaling optimization aspects of the same problem which is session delivery to Mobile IPv6 hosts, or incoming session delivery. We first deal with the signaling optimization of address resolution and paging in IPv6 networks. Secondly, we develop the architectural guidelines and a protocol for end-to-end IP paging in order to improve the robustness of Mobile IPv6 against home agent failure, with a minimum binding update signaling cost. Finally, we deal with the signaling security problems of session delivery. Below we overview each of these proposals in turn.

Hash-based optimizations

We first develop a hash-based broadcast technique using Bloom filters. This allows the network to broadcast some queries such as address resolution or paging request, to multiple hosts, but using a standard-sized single message. In address resolution and IP paging, this procedure saves

bandwidth in each cell, since one message is broadcast instead of many. In IPv6, the destination identifier found in a broadcast query will be 128-bit long. We note that a 128-bit Bloom filter can represent several IPv6 address instead one. We separately explore the benefits of this approach in IPv6 address resolution **(Part 1)** and in network-based IPv6 services **(Part 2)**.

Adaptive IP paging architecture

In the context of our work on network-based IP paging **(Part 2)** we also develop an adaptive IP paging architecture. Adaptive IP paging has been previously proposed, for optimizing the IP mobility management signaling. Our goal is to develop a scalable architecture and algorithm for the configuration of variable sized paging areas, needed by adaptive IP paging.

End-to-end IP paging

We note that "end-to-end IP paging" is possible and it has a very different motivation than network-based IP paging service **(Part 3)**. Our basic goal is to cope with home agent failure or unavailability in Mobile IPv6. We replace the paging area concept by *living area* and argue that, using end-to-end IP paging a mobile host can be mostly located without home agent service. In end-to-end paging the session initiating node -itself- pages the destination mobile host in its living area. End-to-end paging is likely to succeed because most of the mobile users have a fixed living area which they rarely leave and some mobile hosts are not supposed to leave their living area.

Home agent SYN cookies

In the final part **(Part 4)** of the thesis, we address the problem of mobile host and bandwidth protection from denial-of-service. Basing our work on the recent advances in the MIPv6 security area, we argue that IPsec (IP security) cannot possibly protect the mobile hosts from any malicious Internet node. Then, we propose adapting an already existing technique for transparent and cheap mobile host protection with some limitations. SYN-cookies are currently used to protect the public servers from SYN-flooding attacks. We show that SYN-cookies can be reasonably used for transparent mobile host, and network protection against denial-of-service and bandwidth attacks.

Part I

Hash-based address resolution in IPv6

Chapter 2

Review of IPv6 and Neighbor Discovery

Contents

2.1 IPv6

The unexpected success of IPv4[1] resulted in billions of computers connected to a world-wide network of networks called the "Internet". The designers of IPv4 had made provisions for an address space that is considered insufficient today. IPv4 addresses are 32-bit long, which provides 2^{32} addresses, four billion addresses which are not distributed as a function of human population. It is expected that many countries will run out of IP address space in a near future.

IPv6[2] defines 128-bit IP addresses in order to solve the address space problem in all planet. A 128-bit IPv6 address allows up to 2^{128} IP nodes to connect to the Internet, in theory. The designers of IPv6 had two possibilities: keeping everything like IPv4 except the address space, or taking the opportunity of a new design. They chose the second one. IPv6 provides not only a larger address space, but it also incorporates the result of years of internetworking experience and lessons into the new IP protocol. The changes from IPv4 to IPv6, in addition to 128-bit addresses, include: more levels of addressing hierarchy, simpler address auto-configuration, header format simplification, more scalable multicast, more flexible IP header options, anycast addresses, quality-of-service capabilities, and mandatory authentication, data integrity and confidentiality services.

2.1.1 IPv6 addresses

An IPv6 unicast address refers to a single interface[3]. A node may have multiple interfaces, and hence multiple unicast addresses. A node's any unicast address may be used as an identifier

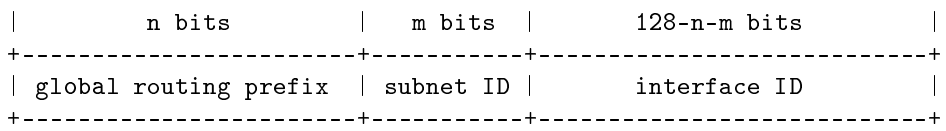


Figure 2.1: Format of unicast IPv6 address.

for that node. The general format for IPv6 global unicast addresses is shown in Figure 2.1. The global routing prefix is a value assigned to a site, the subnet ID is an identifier of a link within the site. The global routing prefix and the subnet ID, together, is called a *subnet prefix*. The interface ID, is the interface identifier. Interface identifiers in IPv6 unicast addresses are used to identify interfaces on a link and they are required to be unique within a subnet prefix. They may also be unique over a broader scope. Interface IDs are required to be 64 bits long (hence, m+n=64) and to be constructed in Modified EUI-64 format. Modified EUI-64 format based interface identifiers may have global scope when derived from a global token (e.g., IEEE 802 48-bit MAC or IEEE EUI-64 identifiers [4]) or may have local scope if a global token is not available. One bit (the u bit) of the 64-bit interface identifier is reserved for distinguishing global/local scope identifiers. Another bit (the g bit) is reserved for distinguishing individual identifiers from group identifiers. As a consequence, in theory, up to 2^{62} interfaces can connect to the same subnet.

IPv6 also supports 128-bit anycast and multicast addresses. There are no broadcast addresses in IPv6. A packet sent to a unicast address is delivered to the interface identified by that address. An anycast address identifies multiple interfaces typically belonging to different nodes. A packet sent to an anycast address is delivered to one of the interfaces identified by that address (in general to the topologically nearest node). A multicast address is an identifier for a set of interfaces typically belonging to different nodes. A packet sent to a multicast address is delivered to all interfaces identified by that address.

2.1.2 Address autoconfiguration

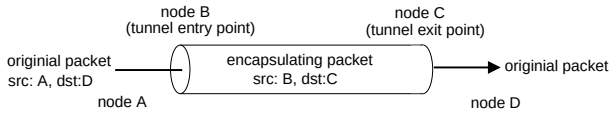
IPv6 defines both stateful and stateless address autoconfiguration mechanisms.

Stateless autoconfiguration does not require any manual configuration of hosts, nor special servers. The stateless mechanism allows a host to generate its own IPv6 address by combining subnet prefix information advertised by routers and an interface identifier that uniquely identifies an interface on a subnet. Before assigning a new address to an interface, a host performs *Duplicate Address Detection* (DAD), in order to avoid duplicate addresses[5]. A candidate IPv6 address on which the DAD procedure is applied is called a tentative address. The procedure uses the neighbor solicitation and neighbor advertisement messages defined by the neighbor discovery protocol (please see Section 2.2 for more details). DAD is performed by multicasting a neighbor solicitation message with a target address field that carries the tentative address. If a neighbor advertisement message is received from the tentative address, the address is already in use by another node and another interface identifier must be generated.

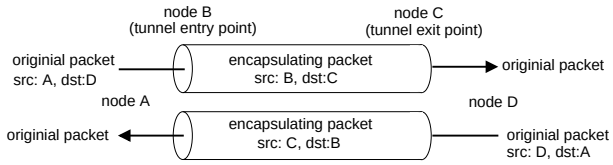
In the stateful autoconfiguration model, hosts obtain interface identifiers and/or other configuration information from a server, which maintains a database and keeps track of which addresses have been assigned to which hosts[6].

2.1.3 IPv6 tunneling

IPv6 tunneling is a technique for carrying an IPv6 packet as a payload of an IPv6 packet[7]. The technique is used by MIPv6 and IPsec. An IPv6 tunnel appears as a point to point link on which IPv6 acts like a link layer protocol. One of the tunnel end-points, called tunnel entry-point,



(a) Uni-directional tunneling.



(b) Bi-directional tunneling.

Figure 2.2: IPv6 tunneling.

encapsulates the original packets received from other nodes or that it generates and sends through the tunnel. The other tunnel end-point, called tunnel exit-point, decapsulates the received packet to obtain the original ones, and routes them their destinations.

An IPv6 tunnel is unidirectional, i.e. packets flow in one direction between the IPv6 tunnel entry-point and exit-point nodes as shown in Figure 2.2-a. Bi-directional tunneling is achieved by merging two unidirectional mechanisms in opposite directions as shown in Figure 2.2-b.

2.2 IPv6 Neighbor Discovery

In IPv6, nodes on the same link use the *Neighbor Discovery* protocol to discover each other's presence, to determine other nodes' link layer addresses, find routers and to maintain reachability information about paths to active neighbors[8]. IPv6 neighbor discovery is defined as part of ICMPv6[9].

2.2.1 Address resolution

In neighbor discovery, a node learns the link layer address of another node (that is assumed to be directly attached to the local link) using a mechanism called *address resolution*. The *Neighbor Solicitation* is an ICMP message as shown in Figure 2.3 is used for address resolution. The *Target Address* is the IPv6 address being resolved. The neighbor solicitation message is sent to a *solicited-node multicast address* specified by the target address. If the target node is indeed present, its network interface should be listening to the corresponding link layer multicast address, and receive the multicast neighbor solicitation message. Upon receipt of a neighbor solicitation message, the target node returns its link layer address in a unicast *Neighbor Advertisement*, which is an ICMP message.

A *Neighbor Cache* entry is created for a target IPv6 address that is being solicited. Upon receipt of a neighbor advertisement reply, the target node's link layer address is recorded in the neighbor cache entry. The address resolution procedure is illustrated in Figure 2.4. In order to limit the storage needed for the neighbor caches, a node may garbage-collect old entries. [8] suggests that implementations should insure that sufficient space is always present to hold the working set of active entries. A small cache may result in an excessive number of neighbor discovery messages if entries are discarded and rebuilt in quick succession. Policies that remove entries that have not been used in some time (e.g., ten minutes or more), are recommended.

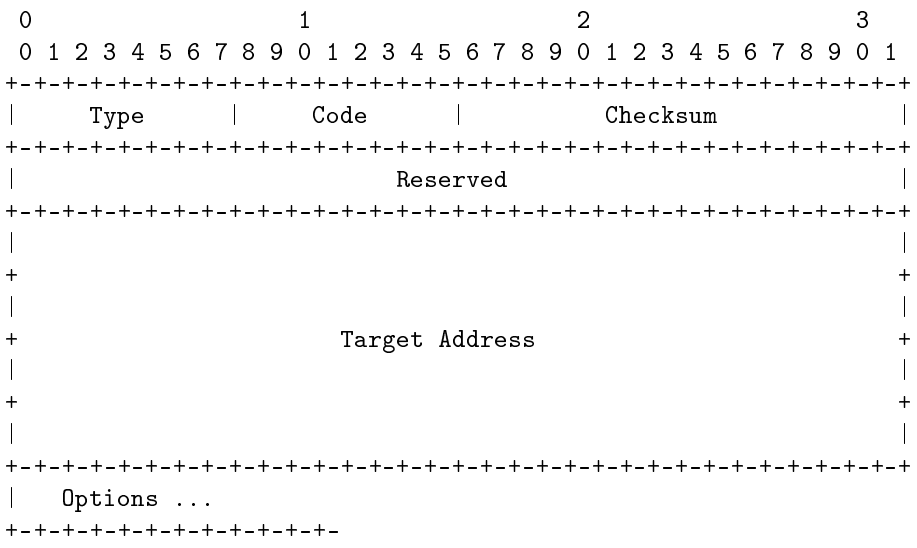


Figure 2.3: Standard neighbor solicitation message.

2.2.2 Unreachability detection

Unreachability detection is a mechanism used for detecting the failure of a neighbor or the failure of the forward path to the neighbor. This improves the robustness of packet delivery in the presence of failing routers, partially failing or partitioned links and nodes that change their link layer addresses. A neighbor cache entry that has not been refreshed for a some time is not deleted. However, its status is suspect. Thus, before sending packets to the subject node, unicast neighbor solicitation probes are sent to verify its reachability.

A neighbor cache entry may be in one of five states:

- *INCOMPLETE*: Address resolution is in progress and the link layer address of the neighbor has not yet been determined.
- *REACHABLE*: The neighbor is known to have been reachable recently (within tens of seconds ago).
- *STALE*: The neighbor is no longer known to be reachable but until traffic is sent to the neighbor, no attempt should be made to verify its reachability.
- *DELAY*: The neighbor is no longer known to be reachable, and traffic has recently been sent to the neighbor. Rather than probe the neighbor immediately, however, delay sending probes for a short while.
- *PROBE*: The neighbor is no longer known to be reachable, and unicast Neighbor Solicitation probes are being sent to verify reachability.

When a node needs to perform address resolution on a neighboring address, it creates an entry in *INCOMPLETE* state and sends a neighbor solicitation message as mentioned above. A neighbor advertisement message sent in response to a neighbor solicitation is referred to as a solicited neighbor solicitation, otherwise it is called an unsolicited neighbor solicitation. The receipt of a solicited neighbor advertisement from a solicited node, confirms that node's reachability and its neighbor cache entry enters leaves the *INCOMPLETE* state for the *REACHABLE* state. Packets can be directly sent to an IP address in *REACHABLE* state.

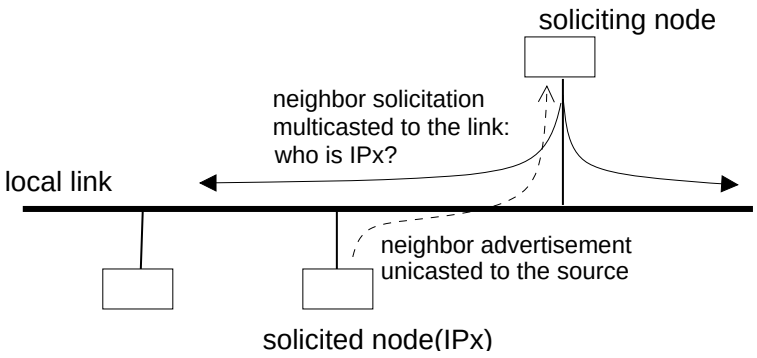


Figure 2.4: IPv6 address resolution.

The neighbor cache entry of an IP address enters the *STALE* state, if packets have not been sent to that address for a longtime. The neighbor cache entry may be valid, but its status is considered suspect. The *STALE* state is useful for quickly verifying the node's reachability when a packet has to be sent to its IP address.

When a packet has to be sent to an IP address in *STALE* state, the packet is sent directly. However, the destination's reachability is not confirmed until a reply packet is received. Hosts may obtain such reachability confirmation upon receipt of a TCP acknowledgment. However, in some cases (e.g. UDP-based protocols or routers) reachability of the destination address cannot be confirmed this way. If a packet that confirms the destination's reachability is not received, the destination address enters the *DELAY* state.

The *DELAY* state is an optimization that gives to upper layers to additional time to receive a reply packet, hence confirm reachability (in cases where such confirmation is possible). If there is no upper layer information that confirms the destination's reachability, or the destination does not reply, its address enters the *PROBE* state. The sending node starts to send unicast neighbor solicitation messages to the destination neighbor until a solicited neighbor advertisement is received, in which case the destination's reachability is confirmed and its cache entry enters the *REACHABLE* state. If no response is received after a number of trials, the destination's cache entry is deleted. Subsequent packets sent to this neighbor recreate a neighbor cache entry in *INCOMPLETE* state and address resolution is performed again.

2.2.3 Router discovery

Router discovery is used to locate the neighboring routers, dynamically learn subnet prefixes and other configuration parameters. Routers periodically multicast unsolicited router advertisements that contain subnet prefixes and other information that allow hosts to autconfigure their IPv6 addresses. Mobile host can also determine the presence of routers by multicasting router solicitations, in which case one or more routers may provide the requested information using solicited router advertisements.

Chapter 3

Review of Bloom filters

Contents

3.1 Introduction

A Bloom filter[10], invented by Burton BLOOM in 1970, is a simple space-efficient randomized data structure for succinctly representing a set, and supporting membership queries. Bloom filters have been used in database applications (e.g. [11]), and recently received attention in the networking area. In this chapter, we present Bloom filters, review some mathematical analysis provided by other researchers and review some network applications of Bloom filters.

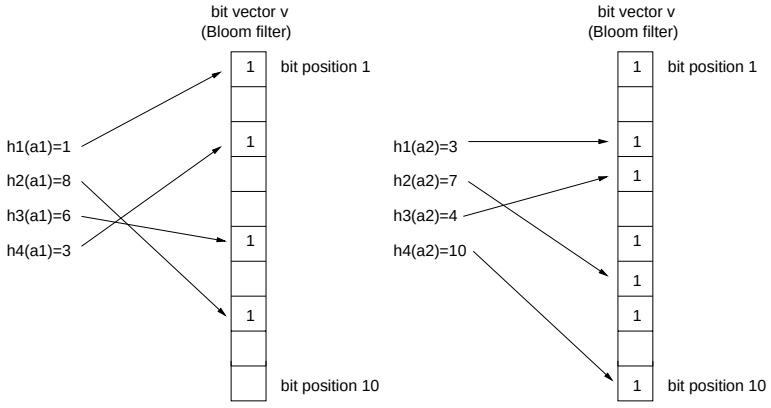
3.2 Bloom filters

A m-bit Bloom filter can be used to represent all elements of a set $A = \{a_1, a_2, ..., a_n\}$, in order to support membership queries. The procedure requires k independent uniform hash functions, $h_1(), h_2(), ..., h_k()$ (where, $1 \leq h() \leq m$). First, all bits of a bit vector v (called Bloom filter) of m bits are set to 0, then for each element $a_i \in A$, the bits at positions $h_1(a_i), h_2(a_i), ..., h_k(a_i)$ in v are set to 1 (a particular bit may be set multiple times). Figure 3.1 illustrates this procedure with 2 elements and 4 hash functions.

The resulting vector v represents all members of A. To check if $b \in A$, the bit positions $h_1(b), h_2(b), ..., h_k(b)$ in A are computed. If any of them is 0, then b is certainly not an element of A. Otherwise, $b \in A$. However, there is a small probability that all bits at positions $h_1(b), h_2(b), ..., h_k(b)$ are set but $b \notin A$. This is called a *false positive*.

3.3 The mathematics

The false positive probability depends on the number of available bits m, the number of elements inserted to the Bloom filter and the number of hash functions. The following analysis is adapted from[12].



(a) First element inserted into the Bloom filter.

(b) Second element inserted into the Bloom filter

Figure 3.1: Bloom filter example.

After inserting n elements to a bit vector of m bits, the probability that a particular bit is still 0 is $(1 - 1/m)^{kn}$. The probability of false positive in this situation is

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

and minimized for

$$k_{opt} = (ln2) \times \frac{m}{n} \tag{3.1}$$

in which case it becomes

$$F = (0.6185)^{m/n} \tag{3.2}$$

With an optimal number of hash functions, the false positive probability will still increase with n, however good performance can be expected if m is large.

3.4 Example network applications of Bloom filters

In this section we present several network application of Bloom filters. A complete survey can be found in [13].

3.4.1 Flow classification

SFB (Stochastic Fair Blue) is a technique for enforcing fairness among a large number of flows[14]. SFB scalably detects and rate limits non-responsive flows through the use of a marking probability derived from the BLUE queue management algorithm and a Bloom filter.

SFB is a simple modification of the BLUE algorithm. BLUE is a queue management algorithm which uses a single marking probability to manage congestion. Other active queue management algorithms rely on queue lengths as an estimator of congestion. However, queue length gives the very little information about the severity of congestion, i.e. the number of competing connections on a link. A single source that transmits at a rate greater than link capacity can cause a queue to build up, just as a large number of sources can. BLUE, on the other hand, performs queue management based on packet loss and link utilization rather than on queue Lents. BLUE maintains a probability used to mark (or, drop) packets when they are queued. If the queue is continually

dropping packets, BLUE increments the marking probability, and hence the rate at which it sends back congestion notification. Conversely, if the queue becomes empty or if the link is idle, the marking probability is reduced. This algorithm allows BLUE to learn the correct rate at which congestion notifications should to be sent back.

SFB combines BLUE and Bloom filters in order to detect non-responsive flows. Each packet is hashed into k bits in a Bloom filter, based on the source-destination pair, or flow label. As a consequence, all packets in a flow hash to the same bits. Each bit i in the Bloom filter has an associated marking probability p_i (where, $0 \leq p_i \leq 1$. When a packet arrives, the marking probability of the associated k bits are increased. If the number of packets that map to a bit goes above a threshold, p_i for that bit is increased. If the number of packets drops to zero, p_i is decreased. The idea behind SFB, is that a non-responsive flow quickly drives p_i to 1 in all of its associated k bits. Responsive flows may share one or two bits with some non-responsive flows. However, unless the number of non-responsive flow is very large, a responsive flow is likely to be hashed into at least one bit that is not polluted by non-responsive flows, thus is has a normal p_i value. The decision to mark a packet is based on p_{min}, which is the minimum p_i of value of all k bits associated with a flow. If $p_{min} = 1$, the packet probably belongs to a non-responsive flow.

In SFB case, a false positive leads to misclassification of a well-behaving flow. There is small probability that, a responsive flow will be punished even though it responds to congestion correctly. However, false positive probability can be made very small if the Bloom filter size is large.

3.4.2 Web cache sharing

Web cache sharing is an important optimization to reduce Web traffic. Proxy caches on the same side of a common bottleneck link cooperate and serve each other's misses. The ICP (Internet Cache Protocol) employs this technique. Nevertheless, the wide deployment of web cache sharing is hindered by the overhead of ICP protocol. In ICP, a proxy multicasts a query to all other proxies whenever a cache miss occurs. As a consequence, as the number of proxies increases, the communication and processing overhead becomes excessive.

In [12], the authors develop a technique called "Summary Cache". Each proxy periodically broadcasts a Bloom filter that represents a summary of URLs of the cached documents, which is stored by other proxies. When a user request misses in the local cache, the proxy checks the stored summaries to determine if the requested document is available in other proxies. If the search succeeds the proxies sends a request to the relevant proxy to obtain the document. Otherwise, the proxy sends a request directly to the Web server. If a proxy wishes to determine if another proxy has a Web page in its cache, it checks the appropriate Bloom filter. False positives and false negatives may occur. False positives are not necessarily caused by Bloom filters, since cache contents may change between periodic updates. However, the small overhead of false positives introduced by Bloom filters in negligible compared to the obtained network bandwidth gain achieved by using a succinct Bloom filter instead of sending the full list of cache contents.

3.4.3 Scalable multicast forwarding

The task of a multicast forwarding engine is forwarding packets along branches of a distribution tree. To achieve this, a router maintains a list of interfaces that connect to each distribution tree. A packet is forwarded by finding the interface list corresponding to the group. If the packet arrived on an interface to the list, it is forwarded over the remaining interfaces of the list.

[15] proposes using Bloom filters to reduce the storage cost of maintaining interface lists. Instead of maintaining a list of interfaces for each group, for each interface a Bloom filter is maintained. If a group is active with respect to an interface, then the corresponding Bloom filter is inserted the group.

False positives in this case, lead to some packets being forwarded incorrectly, which will be eventually discarded by other nodes.

Chapter 4

Hash-based IPv6 address resolution

Contents

4.1 Introduction

In this chapter, we develop an extension to IPv6 Neighbor Discovery, that we call Hash-Based Address Resolution (HBAR). Neighbor Discovery defines an address resolution mechanism to allow a node to dynamically learn and cache the link layer address of other nodes that are attached to it's local link. However, only one address can be resolved at a time. We propose using HBAR when a larger number of addresses need to be resolved concurrently or in quick succession. The 128-bit target address field of a neighbor solicitation message is replaced by a 128-bit Bloom filter.

HBAR is considered useful for routers that need to perform address resolution at a high rate. Hosts may also perform HBAR if upper layers send packets to different IPv6 nodes in the same subnet and in quick succession.

4.2 Protocol description

When HBAR is performed, the 128-bit target address found in a neighbor solicitation message is replaced by a 128-bit Bloom filter (i.e. $m = 128$). The new neighbor solicitation message content is illustrated in Figure 4.1. The B bit indicates hash-based address resolution. The k value is the number of hash functions to be used by receiver nodes for membership query.

The 128-bit Bloom filter (target address field) is calculated by the soliciting node, as follows:

A set $IP_1, ..., IP_i, ..., IP_n$ of IP addresses found in the address resolution queue, are coded by applying k hash functions $h_1, h_2, ..., h_k$ to each IP address ($1 \leq h() \leq 128$). First, all bits of the target address field are set to 0, then for each element IP_i in the queue, the bits at positions

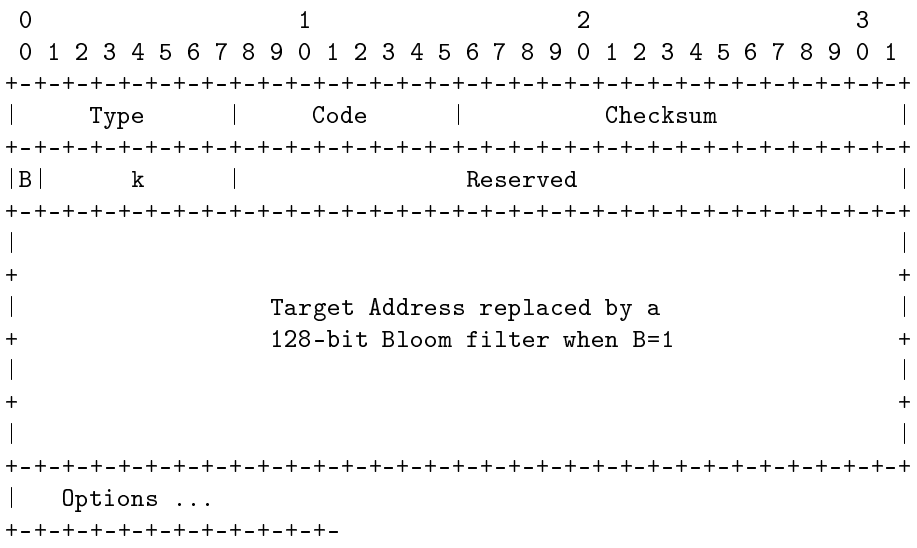


Figure 4.1: Hash-based neighbor solicitation message.

$h_1(IP_i), ..., h_k(IP_i)$ in the target address field are set to 1 (a particular bit may be set multiple times). Note that a total of $n \times k$ hashes are computed. The hash functions are found in a table $h_1, h_2, ..., h_{k_{max}}$. The optimal number of hash functions corresponding to n is also found in a table similar to Table 4.1. The resulting message is sent to all-nodes multicast address. Please see Section 4.4.2 for detailed discussion of using the all-nodes multicast address.

Upon receipt of the neighbor solicitation message, a node IP_x checks the k bit positions $h_1(IP_x), h_2(IP_x), ..., h_k(IP_x)$ of the received Bloom filter (the results of hash functions are computed only once upon configuring an IPv6 address). If any of them is 0, then IP_x is certainly not being solicited. Otherwise, IP_x is being solicited and should respond with a neighbor advertisement message. Note that the hash functions are well-known. I.e. the soliciting node and the receiver nodes apply the same hash functions for Bloom filter construction and membership query, respectively.

The HBAR procedure is illustrated in Figure 4.2.

4.3 False neighbor advertisements

HBAR introduces a small false neighbor advertisement risk. A node IP_y may receive a hash-based neighbor solicitation message with set target address bits at positions $h_1(IP_y), h_2(IP_y), ..., h_k(IP_y)$, although its address is not being solicited. In this case the node IP_y will send an unnecessary neighbor advertisement. We call this a "false neighbor advertisement".

The false neighbor advertisement probabilities depend on the number of concurrently resolved IPv6 addresses. This is shown in Table 4.1.

4.4 Analysis

4.4.1 Bandwidth gain

By performing HBAR, a node multicasts 1 neighbor solicitation instead of multicasting n neighbor solicitations at the cost of $N \times F$ false neighbor advertisements, where N is the number of hosts in the subnet. The bandwidth gain in the core network is:

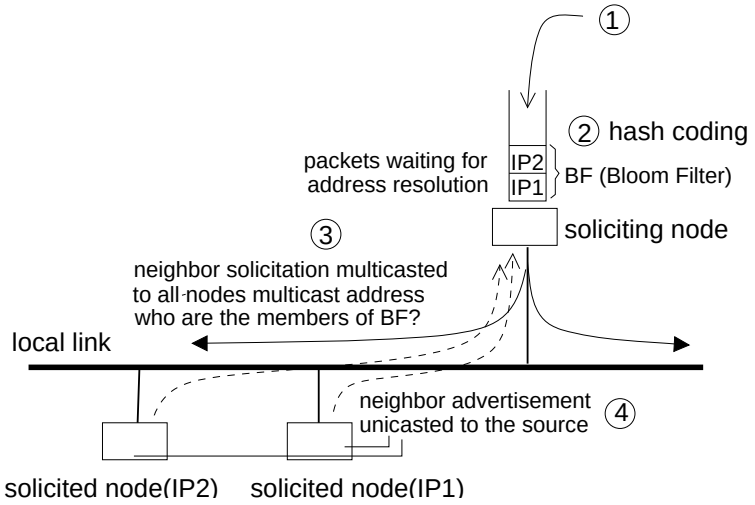


Figure 4.2: Hash-based IPv6 address resolution. In this example $n = 2$.

n	k_{opt}	F
2	44	0.000000
3	29	0.000000
4	22	0.000000
5	17	0.000005
6	14	0.000035
7	12	0.000153
8	11	0.000459
9	9	0.001078
10	8	0.002134
11	8	0.003732
12	7	0.005947
13	6	0.008821
14	6	0.012367
15	5	0.016575

Table 4.1: The optimal number of hash functions k that minimizes the false neighbor advertisement probability F. The k and F values are set according to well-known equations Equ.3.1 and Equ.3.2 with $m = 128$.

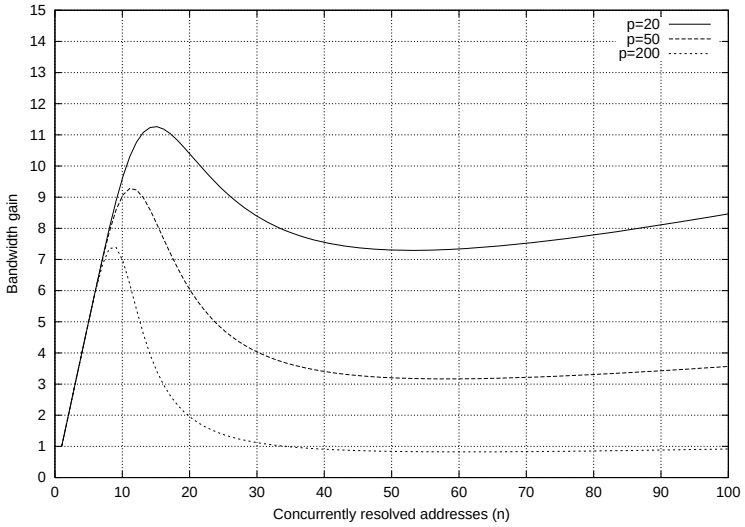


Figure 4.3: Bandwidth gain per cell.

$$G_{core} = \frac{n}{1 + N \times F} \tag{4.1}$$

In a subnet with access points, the false neighbor advertisement rate in each cell depends on p, the number of hosts per cell. As a consequence, the bandwidth gain in each cell is:

$$G_{wireless} = \frac{n}{1 + p \times F} \tag{4.2}$$

Figure 4.3 shows the bandwidth gain per cell, for different cell sizes. HBAR becomes more attractive as cell sizes become smaller. The n that we choose must maximize the bandwidth gain as much as possible while $F \simeq 0$, so that false neighbor advertisements have negligible energy consumption overhead. This roughly corresponds to the interval where $\frac{dG}{dn} \simeq 1$ in Figure 4.3, i.e. $G_{wireless} \simeq n$.

Without justification we are satisfied with $n \leq 9$, since the false neighbor advertisement probability F is roughly $\frac{1}{1000}$. Please see Section 4.5 for further discussion on bandwidth gain.

4.4.2 Energy gain

The standard neighbor solicitation message is sent to a *solicited-node multicast* address that is chosen as a function of the target IPv6 address that is being resolved[8]. A solicited-node multicast address is formed by a host by taking the low-order 24 bits of its unicast IPv6 address and appending these bits to a well known prefix[3]. Then, the host instructs its network interface card to accept packets sent to the corresponding link layer multicast address. Thus, it is not interrupted by neighbor solicitation messages targeted to other nodes. CPU cycles and energy (on mobile devices) are saved, which is the design rational behind the standard address resolution[16]. In HBAR, a neighbor solicitation message is sent to the *all-nodes multicast address*, since the low-order 24 bits of the concurrently resolved (random) IP addresses are unlikely to be the same. A packet sent to the all-nodes multicast address will trigger an interrupt on every node attached to the link. This will unnecessarily force every node in the subnet to perform membership check to the Bloom filter by comparing two 128-bit strings (there is no hash computation cost on receiver nodes).

While the above analysis is true, with the introduction of HBAR it becomes incomplete. Let $drop_2$ and $drop_3$ the energy cost due the reception of an unnecessary multicast packet and dropping

it, at the link layer (layer 2) and the IP layer (layer 3) respectively. Let $decap_2$ the energy consumed for passing a link layer packet to the IP payer, i.e. the decapsulation overhead. Upon receipt of a neighbor solicitation message destined for another IP host, the solicited-node multicast optimization saves $decap_2 + drop_3$.

On the other hand HBAR reduces the rate of address resolution events in the subnet and hence the rate of link layer multicast packets received by on-link nodes, by a factor of n. The packet processing gain can be roughly defined as:

$$G_e = \frac{n \times drop_2}{decap_2 + drop_3}$$

The above formula does not take into account the energy consumed by the receiver circuit. We know that this is an important source of battery drain. Using HBAR every receiver in the subnet receives 1 address resolution packet in stead of n. In 802.11, for example, terminals can enter a power saving more (PSM) in which their receiver circuit is mostly turned off. Incoming frames, including multicast frames, are buffered by the access points. For each buffered multicast frame, the terminal in PSM must send a poll message to its associated access point in order to retrieve the buffered frame, and also turn on its receiver circuit[17]. HBAR can reduce the number of frames buffered by the access points, hence reduce the burden of receiving the multicast packets for address resolution. We hope to evaluate this issue in the future.

4.5 Address resolution policies

4.5.1 Threshold controlled mode (TCM)

In TCM mode, our goal is to use HBAR only when necessary and thereby reducing the link layer interrupt rate. In this case the soliciting node will perform HBAR when the number of packets waiting for address resolution reaches or exceeds a threshold

$$n \geq 9$$

and 9 addresses will be concurrently resolved at a time. Otherwise standard address resolution will be performed, which avoids unnecessary link layer interrupts.

In this case the hash computation cost per HBAR will be 81 hashes ($k_{opt} = 9$ hash functions applied to $n = 9$ IPv6 addresses). Let $N = 5,000$ and $p = 20$, the bandwidth gain per HBAR will be $G_{core} = 1.4$, $G_{wireless} = 8.54$. Note that, depending on the subnet size, HBAR may result in bandwidth loss in the core network, but bandwidth gain in wireless links. Assuming $N = 10,000$, we have $G_{core} = 0.76$, $G_{wireless} = 8.54$.

4.5.2 Upper bounded mode (UBM)

In UBR mode our goal is to reduce the number of link layer multicast packets received by on-link nodes. In this case, HBAR will be performed whenever the number of packets waiting for address resolution is

$$n > 1$$

and up to 9 addresses (i.e. 9 is the upper bound) will be concurrently resolved per HBAR. If $n = 1$, standard address resolution will be performed, and a node will not be interrupted for a neighbor solicitation packet targeted at another node.

In this case the hash computation cost per HBAR will be between 88 (2×44) and 81 (9×9) hashes. The bandwidth gain per HBAR will depend on n. Note that bandwidth gain per HBAR is not necessarily smaller than TCM. By limiting the address resolution rate to a low value, it is possible to keep n at an optimal level.

4.6 Example hash function

HBAR requires a set of well-known uniform and independent hash functions to be used by access routers and mobile hosts. In this section, we show an example implementation.

The choice of the hash function implementation is important for reducing the CPU cost at soliciting nodes. Below we show an example implementation that can output 128 independent hashes of an input value. This implementation is suitable for constructing a Bloom filter, since a large number of independent and uniform hash functions are needed (88 in our case).

The algorithm was initially designed for dictionary words represented as byte strings in spell-checkers and known as *Pearson's hash* [18]. An IPv6 address is also represented as a byte string[19]:

```
struct in6_addr {
        unsigned char s6_addr[16];/* IPv6 address */
        }
```

We modify the original algorithm by applying a random permutation of values between 1 and 128, since HBAR requires a hash output between 1 and 128. This permutation is recorded in a table. The same permutation must be used by soliciting and listening nodes.

```
unsigned char T[128]={  5,  89,  14,  97, 103,   9,  77,  38,
                       48,  28,   0,  15,  88,  84,  62,  93,
                       25,  59,  86, 118,  81,  50,  11, 125,
                        2,  12,  22,  91,  96,   8,  40,  34,
                       32,  85,  20,  63,  33,  37,  68, 116,
                       29,  46,  55, 121, 112,  69, 101,  36,
                       76,  53, 117, 122, 119,  17,  58,  65,
                      110,  42,  30, 109,  67,  73,  75, 111,
                       45,  71,  54,  18, 115, 124,  49, 102,
                       72,  21,  74, 106,  82, 123, 108,  27,
                      127,  44,  43,  92,  99,  31, 126,  39,
                      107,   1,   6,  90,  51,  83,  80,  79,
                        7, 120,  13,  47,  66, 113,  35,  57,
                       98,   3,  56,  16,  78,  95,  87,  10,
                       23,  64,  19, 114,  61,  94,  26,  70,
                      105,  60, 100,  52, 104,  24,  41,   4};
```

The modified algorithm is input two arguments: the 16-byte IPv6 address and an index. The index value determines the hash function. For example by setting $index = 0$ the first hash function is applied, by setting $index = 1$ the second hash function is applied etc.

```
unsigned char hash(struct in6_addr addr, int index){
                int i;
                unsigned char h=T[index];
                for(i=1;i<16;i++)
                        h=T[h^addr.byte[i]];
                return h;
                }
```

One octet of the input IPv6 address is input to the hash function at a time and its processing requires only an exclusive-OR (denoted ^) and an indexed memory read.

Chapter 5

Address resolution performance of a wireless access router

Contents

5.1 Introduction

In this chapter, we analyze the address resolution performance of a wireless access router. The subject access router provides Internet access to a subnet with many wireless access points hence many mobile hosts. Every mobile host expect incoming real-time sessions such as VoIP (Voice over IP) and other sessions from correspondent nodes. Incoming packets are possibly routed to their destinations using Mobile IPv6 (MIPv6). Even though Hash-Based Address Resolution (HBAR) is independent of the design of MIPv6, in this chapter we assume that the mobile hosts are capable of performing IP-layer handovers and they receive incoming sessions. Thus we assume that they are MIPv6 hosts. A detailed analysis of MIPv6 falls in the scope of the next part of this thesis.

For our simulations, we have chosen a wireless subnet with many mobile hosts, because we suspect that HBAR is most useful in this case. We assume that MIPv6 hosts are similar to the current cellular terminals, they rarely initiate/receive calls, and they are mostly idle. Although each host rarely receives incoming sessions, we expect that the total incoming session rate will be high.

The *neighbor cache* is at the heart of the address resolution rate. Upon incoming session, the wireless access router may or may not perform multicast neighbor solicitation. Address resolution will be needed if the wireless access router does not already have a neighbor cache entry for the destination host.

A multicast neighbor solicitation message consumes bandwidth in all cells of the subnet. We assume that multicast neighbor solicitation messages are rate limited. This allows us to reap the gain of HBAR. The wireless access router deploys an address resolution queue. IP addresses (for which no neighbor cache entry is found) enter the address resolution queue, and served on a first in first out basis. An incoming session can be established (by end hosts) when the destination IP address resolved, i.e. the first packet of the session is delivered to its destination. A high incoming session arrival rate (to destinations without neighbor cache entry) may increase the number of queued address resolution processes and lead to session setup delays.

We analyze the delays in the address resolution queue deployed by the wireless access router. We analyze the address resolution performance in different situations, with an without HBAR assistance.

5.2 Simulation

We have written a program to simulate 1 hour of incoming session traffic in a subnet. Mobile hosts enter the subnet at a rate of 1 mobile host/second, and stay for a random duration that is uniformly distributed between 60 seconds and 10,000 seconds ($\sim$ 3 hours). Before the measurements, the simulation starts with a stabilization phase. During this phase, the number of hosts in the subnet converge to the expected value $\frac{10,000-60}{2} \simeq 5,000$. The number of neighbor cache entries in access router memory also converge to an average value that depends on the average incoming session rate I and the neighbor cache lifetime *clt*. A neighbor cache entry that is not used is garbage collected *clt* minutes after its creation.

The multicast neighbor solicitations (standard or hash-based) of the wireless access router are rate limited to NSR (neighbor solicitation rate) neighbor solicitations per second, which is set 1 and 0.25 in different configurations. Please see Section 5.3 more detailed information on the tested configurations.

When a mobile host enters the subnet it sends a MIPv6 binding update to its home agent and receives its acknowledgment. As a consequence, the wireless access router creates a neighbor cache entry for a mobile host that enters its subnet. When the mobile host leaves the subnet, its neighbor cache entry (if not already expired) is "not" deleted. Upon incoming session, the wireless access router performs address resolution if the destination host has not left the subnet, but its neighbor cache entry has expired (garbage collected after *clt* minutes). The cache lifetime *clt* is set to 1, 10, 20 and 30 minutes in different simulations. Incoming sessions are driven by a Poisson process. The average incoming session rate I is set to 1, 2 and 3 incoming sessions per hour, in different simulations. During 1 hour simulation, some hosts receive more than I sessions, while others receive less than I sessions (including zero), however the average is I. Retransmitted packets do not enter the address resolution queue.

The session duration is uniformly distributed between 10 seconds and 5 minutes. Mobile hosts also initiate outbound sessions at an average rate of 1 session per hour. A neighbor cache entry at the access router is created for a host that initiates an outgoing session (if its neighbor cache entry has expired). If a mobile host initiates or receives a session and already has a neighbor cache entry, its neighbor cache entry is refreshed.

In summary, the following parameters are input to the simulation:

- Neighbor cache lifetime *clt* (seconds).
- Incoming session rate per host I (sessions per hour).
- Neighbor solicitation rate NSR (neighbor solicitation per second).

and the following measurements are made during 1 hour simulation that follows the stabilization phase:

- The queueing delay in the address resolution queue of the wireless access router (in seconds).

- The number of neighbor cache entries that are created and maintained by the wireless access router.
- *ips* (interrupt per second). The average rate of link layer interrupts caused by HBAR (per mobile host).
- *pps* (packet per second). The average rate of link layer multicast packets generated by the access router.

5.3 Tested configurations

- Standard address resolution (denoted HBAR:OFF) and NSR=1. With this configuration $ips = 0$, $pps \leq 1$.
- HBAR in TCM mode (denoted HBAR:TCM) and NSR=1. With this configuration $ips \leq 1$, $pps \leq 1$.
- HBAR in UBM mode (denoted HBAR:UBM) and NSR=1/4. With this configuration $ips \leq 1/4$, $pps \leq 1/4$. In terms of queueing delay, HBAR:UBM outperforms HBAR:TCM with the same NSR setting since HBAR:UBM is not threshold controlled and freely used. Therefore, in UBM mode we test a lower NSR.

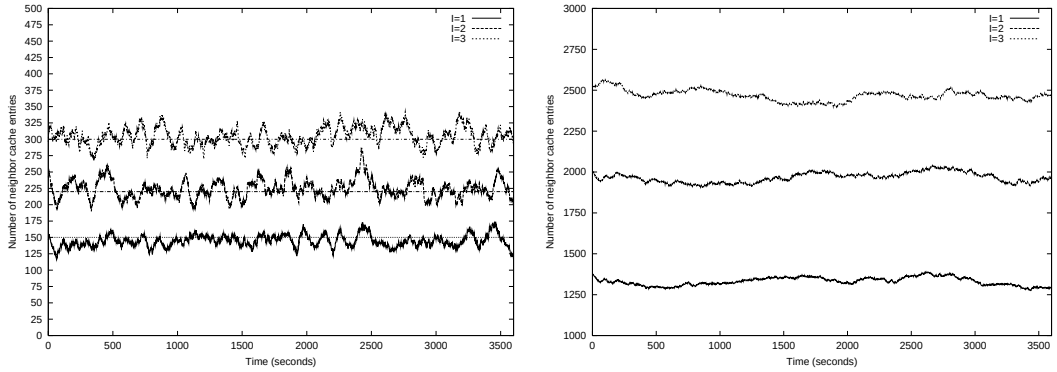
5.4 Constant incoming session rates

In the following simulations the session arrival rate is set to $I = 1$, 2 or 3 and does not change during the simulated hour. As expected, during the stabilization phase that precedes the simulation, the number of cache entries in access router memory converge to a value that depends on I and *clt*. The approximate number of cache entries in access router memory is shown in Table 5.1. Figure 5.1 shows the exact values for $clt = 1min$ and $clt = 10min$.

Number of neighbor cache entries	$I = 1$	$I = 2$	$I = 3$
$clt = 30min$	3500	4500	5200
$clt = 20min$	2500	3500	4000
$clt = 10min$	1350	2000	2500
$clt = 1min$	150	225	300

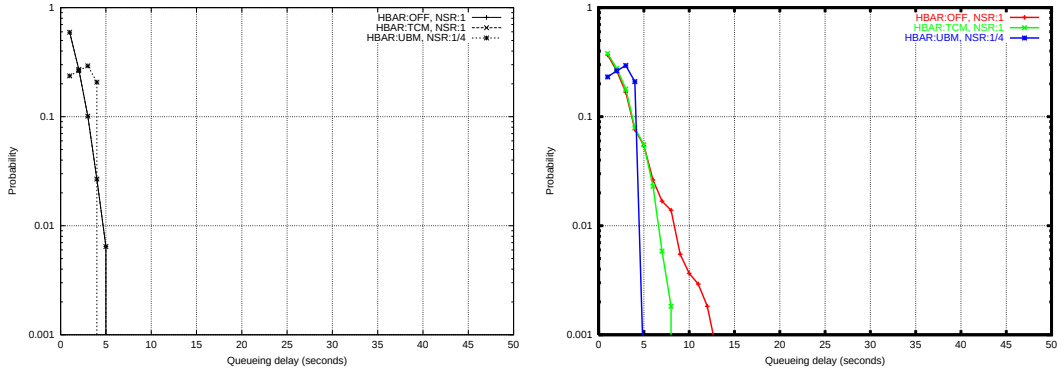
Table 5.1: Observed number of cache entries in access router memory.

In the first simulation (Figure 5.2-a) $clt = 30min$. I.e. a neighbor cache entry that is not used during 30 minutes is garbage collected. The average incoming session arrival rate is set to $I = 1$. In these conditions, the number of neighbor cache entries in the access router memory converged to ~ 3500 and standard address resolution performed well. In TCM, the HBAR threshold ($n = 9$) was never reached, and HBAR was not performed. As a consequence, we obtained the same result with the HBAR:OFF configuration. UBM performed slightly better, although neighbor solicitation rate was 4 times smaller. Table 5.2 shows the average link layer interrupt rate *ips* and link layer multicast packet rate *pps*. Clearly, in HBAR:OFF the mobile hosts are not interrupted upon neighbor solicitation targeted at other nodes in the subnet, since neighbor solicitation messages are always sent to solicited-node multicast address, hence $ips = 0$. In TCM, with $clt = 30min$, HBAR was not performed and $ips = 0$. In UBM, on-link nodes suffered from $ips = 0.165$ link layer interrupts per second, since neighbor solicitation packets are sent to all-nodes multicast address. However, nodes profited from a smaller rate of link layer multicast packets to be received (and dropped) by their link layer interface: $pps = 0.225$ instead of $pps = 0.560$.

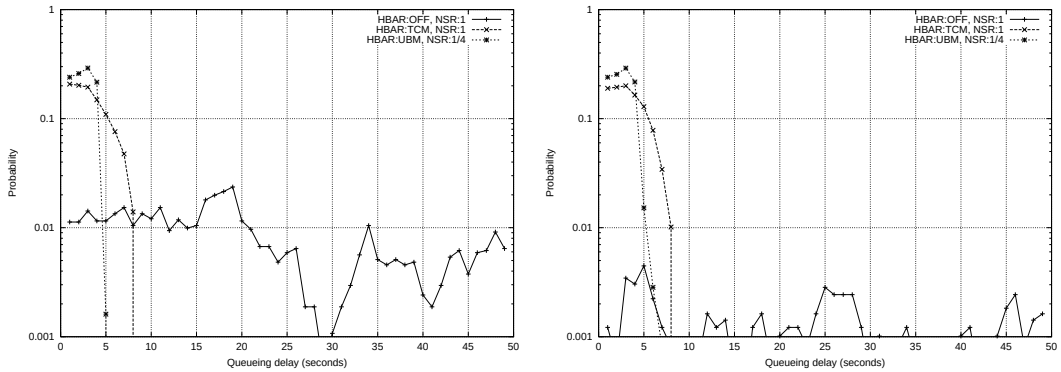


(a) $clt = 1min.$ (b) $clt = 10min.$

Figure 5.1: Number of neighbor cache entries in access router memory.

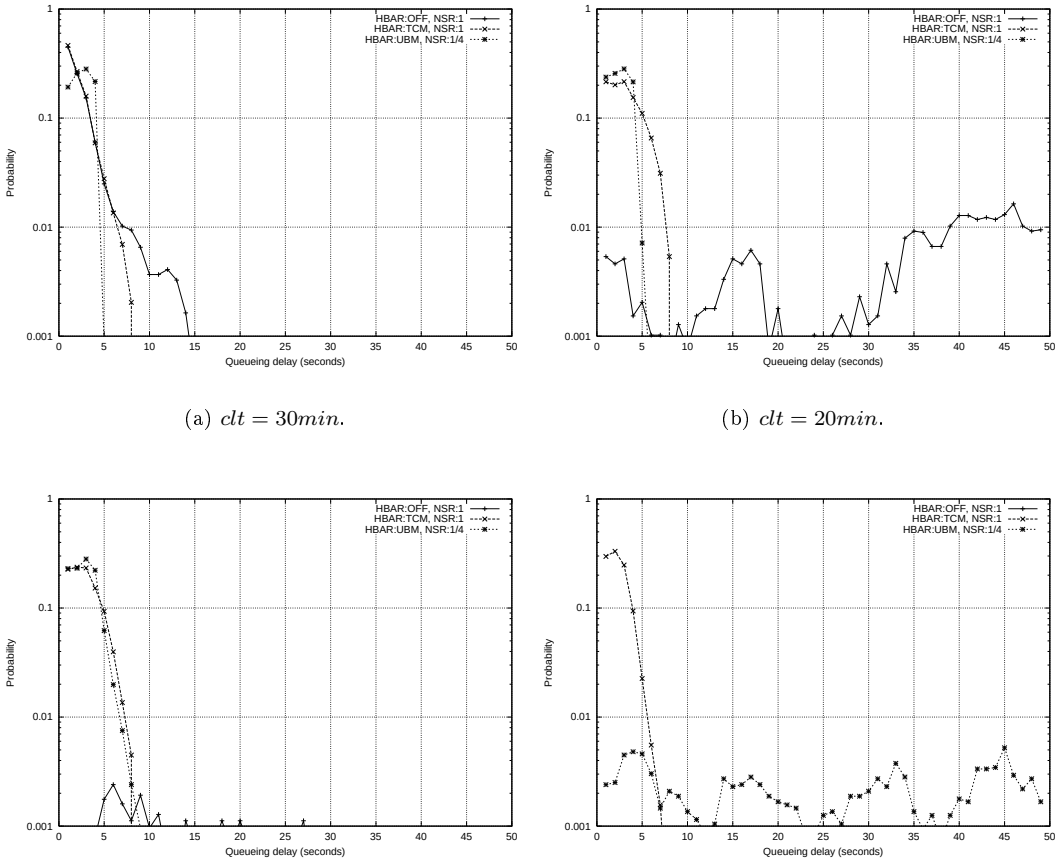


(a) $clt = 30min.$ (b) $clt = 20min.$



(c) $clt = 10min.$ (d) $clt = 01min.$

Figure 5.2: Queueing delays ($I = 1$).



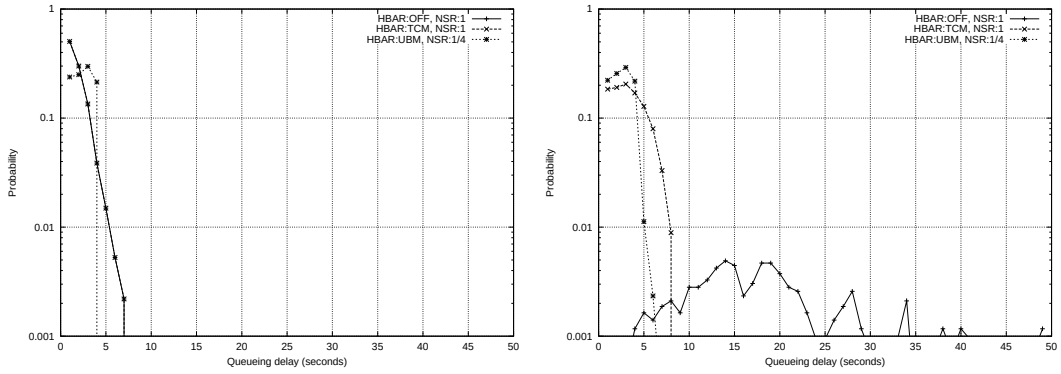
(a) $clt = 30min.$

(b) $clt = 20min.$

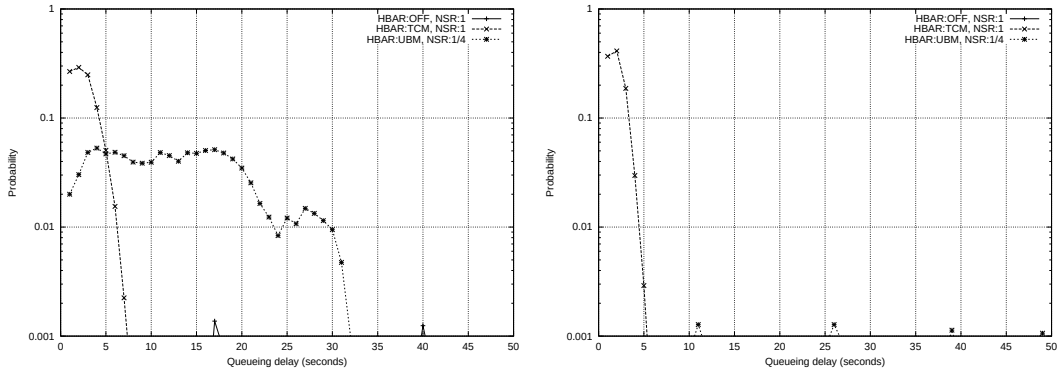
(c) $clt = 10min.$

(d) $clt = 01min.$

Figure 5.3: Queuing delays ($I = 2$).



(a) $clt = 30min.$ (b) $clt = 20min.$



(c) $clt = 10min.$ (d) $clt = 01min.$

Figure 5.4: Queuing delays ($I = 3$).

In Figures 5.2-a-b-c we reduce *clt* and hence, the number neighbor cache entries in access router memory drop. As we decreased *clt*, HBAR:OFF with rate limiting failed and excessive queueing delays occurred (queueing delays more than 50 seconds are not shown). With TCM, the HBAR threshold was reached more frequently and queueing delays were smaller. UBM with NSR=1/4 outperformed TCM with NSR=1. This result is not surprising. In UBM, the address resolution queue is emptied every 4 seconds provided that $n \leq 9$ packets are waiting as described in Section 4.5. In TCM, on the other hand, HBAR is not performed until there are $n \geq 9$ packets waiting in the address resolution queue. For example, 8 packets may enter the address resolution queue, but the 9^{th} packet may not arrive and the 8^{th} packet may wait 8 seconds (since NSR=1).

I=1	ips_{TCM}	ips_{UBM}	pps_{TCM}	pps_{UBM}
$clt = 30min$	none	0.165	0.560	0.225
$clt = 20min$	0.001	0.203	0.760	0.238
$clt = 10min$	0.020	0.231	0.906	0.244
$clt = 01min$	0.048	0.243	0.960	0.247
I=2	ips_{TCM}	ips_{UBM}	pps_{TCM}	pps_{UBM}
$clt = 30min$	0.001	0.187	0.672	0.234
$clt = 20min$	0.020	0.231	0.914	0.245
$clt = 10min$	0.098	0.248	0.976	0.249
$clt = 01min$	0.209	failed	0.991	failed
I=3	ips_{TCM}	ips_{UBM}	pps_{TCM}	pps_{UBM}
$clt = 30min$	none	0.184	0.633	0.233
$clt = 20min$	0.031	0.240	0.946	0.248
$clt = 10min$	0.152	failed	0.984	failed
$clt = 01min$	0.368	failed	0.996	failed

Table 5.2: Link layer interrupt, and link layer multicast rates.

It is important to note that we always have:

$$ips_{UBM} > ips_{TCM}$$

and

$$pps_{UBM} < pps_{TCM}$$

as shown in Table 5.2. I.e., UBM is more expensive in terms of link layer interrupt cost, but it reduces the number of multicast link layer multicast packets received by on-link nodes. With TCM, HBAR is performed only when $n \geq 9$, otherwise standard address resolution is performed. In this case, the neighbor solicitation packets are mostly sent to the solicited-node multicast address and unnecessary link layer interrupts are avoided. In UBM, on the other hand, HBAR is performed whenever $n > 1$, which reduces rate of link layer multicast packets.

In Figures 5.3-5.4, we repeat the same simulations with increased incoming session rates $I = 2$ and $I = 3$. Without HBAR, standard address resolution with rate limiting failed, especially with small *clt*. Note also that UBM with NSR=1/4 outperformed TCM with NSR=1, however it has an upper limit. UBM with NSR=1/4 could not cope with our $I = 2$ and $clt = 1min$ setting and $I = 3$ and $clt = 10min$ (and hence $clt = 1min$) setting. This result suggests that in difficult conditions UBM may be used with a higher neighbor solicitation rate. We have repeated the HBAR:UBM simulation with our most difficult setting, i.e. $I = 3$ and $clt = 1min$, but with an NSR=1 setting this time. UBM kept the queueing delays under 3 seconds, the probability of 3 seconds delay being 0.001.

5.5 Increasing incoming session rates

In the following simulations we set $clt = 30min$. Our goal is to show that a large neighbor cache lifetime does not help during the transition periods where the incoming session rates constantly increase ($\triangle I/\triangle t > 0$). A large neighbor cache lifetime is rather helpful when the incoming session rate is constant, as shown in the previous analysis.

5.5.1 Fast increase

In this section, we simulate a fast increase in session arrival rate. The session arrival rate in the subject subnet suddenly increases.

During the stabilization phase that precedes the simulation, the average incoming session rate is set to $I = 1$. The neighbor cache lifetime is 30 minutes. Thus, the number of cache entries in access router memory converge to ~ 3500 entries as shown in Table 5.1. At the beginning of the one hour simulation, we suddenly set $I = 2$. Table 5.1 shows that ~ 4500 cache entries are normally needed in these conditions. However, the access router has not yet that many cache entries, it has ~ 3500 entries. Thus, we expect that the access router can not quickly adapt to these new conditions and a number of incoming session may suffer from excessive address resolution delays.

The results are shown in Figure 5.5-a. Without HBAR, hundreds of incoming sessions failed. With HBAR assistance, all addresses were resolved within reasonable time limits. UBM outperformed TCM. The number of IP addresses in address resolution queue, without HBAR is shown in Figure 5.5-b. Without HBAR, we observe a sudden increase in the number of queued packets. Clearly, this is due to the fast transition to $I = 2$, and results in increased address resolution delays. Figure 5.5-c show the number of cache entries in access router memory. 2000 seconds (half an hour) have passed before the access router could adapt to the new conditions.

Figure 5.5-d shows the link layer interrupts caused by HBAR:TCM. 10 link layer interrupts occurred during the transition period (which took $\sim$20 minutes).

5.5.2 Slow increase

In this section, we increase the transition time to half an hour. At the beginning of the simulation the average incoming session rate is set to $I = 1$, gradually increases during half an hour, and reaches $I = 2$.

The results are shown in Figure 5.5. Standard address resolution, without HBAR assistance, performed better than the previous simulation. However, a total of 38 session waited more than 10 seconds. With HBAR, all addresses were resolved within reasonable time limits. Again, UMB outperformed TCM.

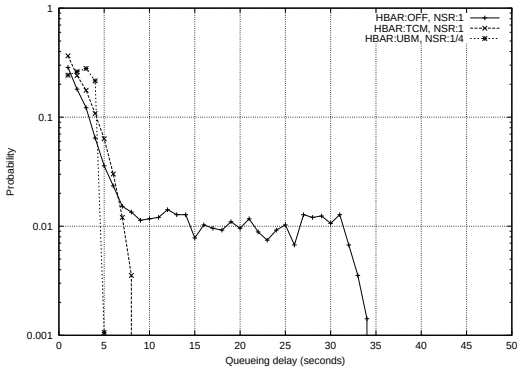
5.5.3 Very slow increase

When we increased the transition time to 1 hour, standard address resolution could resolve all addresses within reasonable time limits. These results are shown in Figure 5.7.

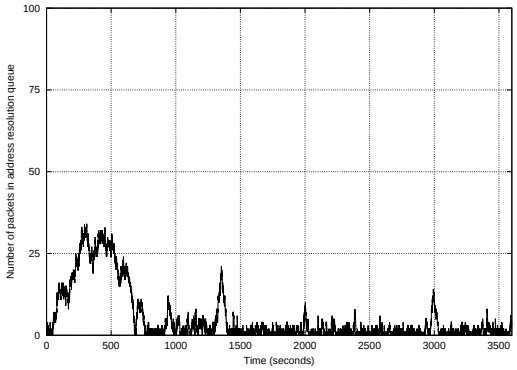
5.6 Address resolution with loss of state

In this secion, we evaluate the address resolution performance of a wireless access router that has lost its neighbor cache state. At a point of time, the access router crashes and reboots with loss of state.

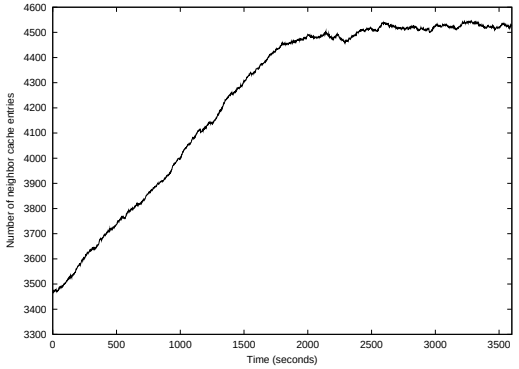
During the period where the router is down, the mobile hosts' binding caches held by their MIPv6 home agents still point to the subject subnet (until they move to another subnet). Thus, correspondent node packets are routed to the subject subnet. The packets that are received during that period are lost. We assume that the router quickly reboots but with loss of state. After rebooting, the router still receives incoming packets from correspondent nodes destined for mobile



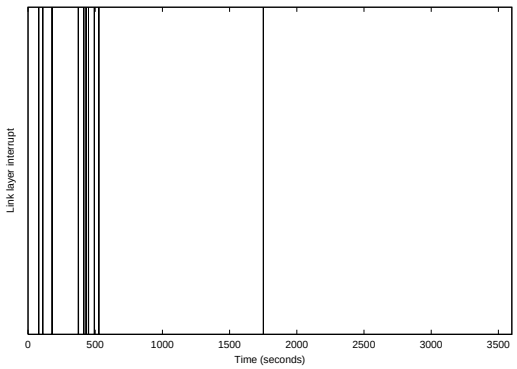
(a) Queueing delays.



(b) Number of packets in address resolution queue (HBAR:OFF).

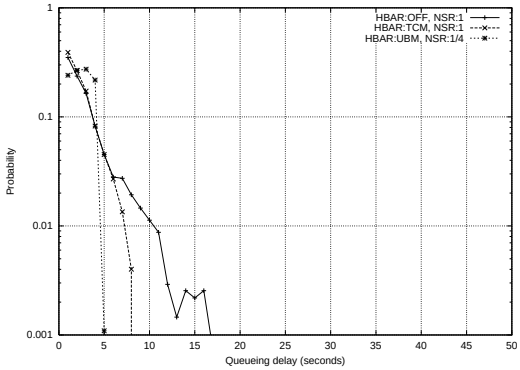


(c) Number of neighbor cache entries.

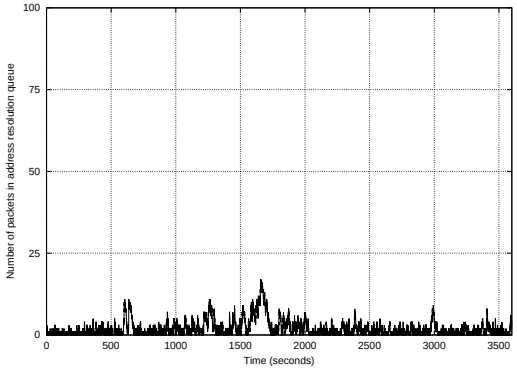


(d) Link layer interrupts (HBAR:TCM).

Figure 5.5: Fast increase.

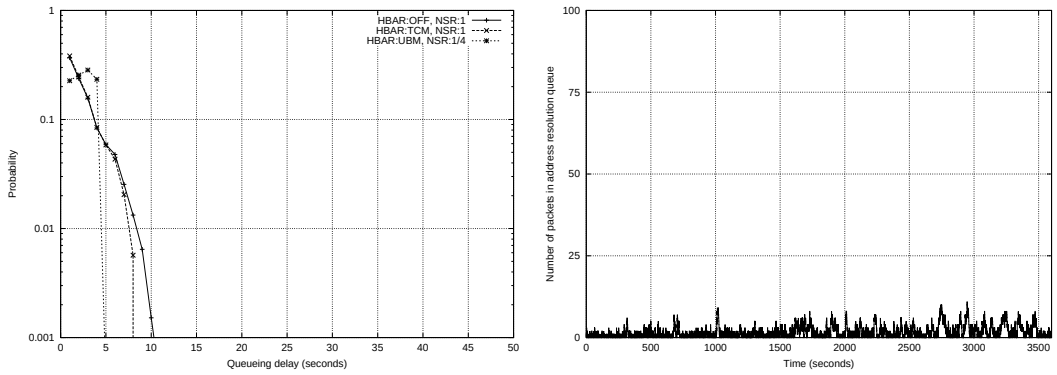


(a) Queueing delays.



(b) Number of queued address resolution processes (HBAR: OFF).

Figure 5.6: Slow increase.



(a) Queueing delays.

(b) Number of queued address resolution processes (HBAR: OFF).

Figure 5.7: Very slow increase.

hosts in its subnet. Having lost its neighbor cache state, the router starts to perform address resolution for every incoming packet destinated for a different mobile host.

5.6.1 Subnet with 5,000 hosts

We first use the same mobility parameters and subnet size than the previous chapter. I.e., there are $\sim 5,000$ users in the subject subnet. However, we set $I = 1$ and $clt = 30min$ which are the best conditions during the uptime of the wireless access router.

In this case, the simulation starts (followed by the stabilization phase) with 0 neighbor cache entries. The results are shown in Figure 5.8. During 1200 seconds (20 minutes) after the access router's reboot with loss of state, denoted "Recovery Period" in Figure 5.8-b, incoming sessions suffered from excessive queueing delays. During this period, almost every incoming packet triggered address resolution. With HBAR, all addresses were resolved within reasonable time limits. In TCM, HBAR was performed 21 times, as a consequence 21 link layer interrupts were caused during the recovery period, i.e. 1 interrupt per minute. As the router's neighbor cache is filled with new entries, the HBAR:TCM frequency, hence link layer interrupt rate felt to its normal level (i.e. ~ 0).

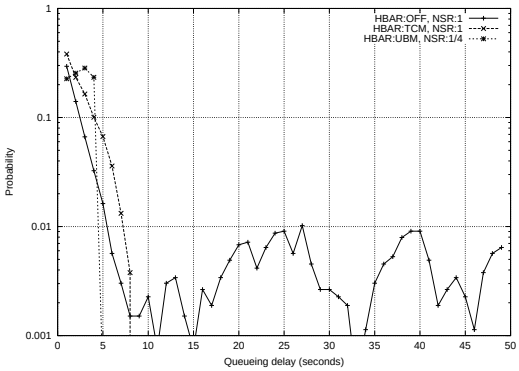
With 5,000 hosts UBM outperformed TCM, the queueing delays with UBM were smaller at higher link layer interrupt cost *ips* (but smaller *pps*).

5.6.2 Subnet with 10,000 hosts

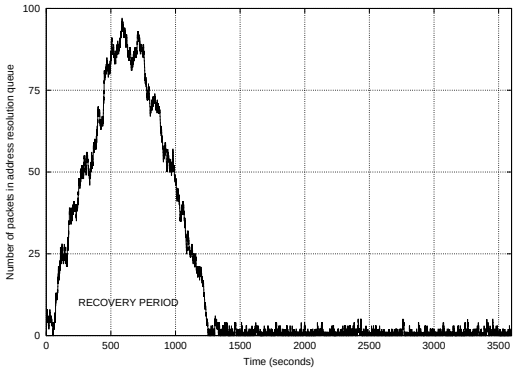
In the following simulations we change the mobility parameters. The maximum time spent in the subnet is increased to 20,000 seconds (~ 6 hours). As a consequence, during the stabilization phase the number of users in the subnet converge to $\sim 10,000$. Thus, we simulate a subnet that covers a wider geogpraphical area, a larger number of access points, and a larger number of users. The neighbor cache lifetime is set to $clt = 60min$. Similarly to the previous simulation, at the begining of the measurements the access router reboots with loss of state. The results are shown in Figure 5.9. Using standard address resolution with rate limiting the recovery period was extended to more than 1 hour in this case. TCM was successful at higher link layer interrupt cost. During the first 100 seconds that followed the crash, the mobile hosts suffered from 0.2 link layer interrupts per second, which gradually decreased as the access router resolved more addresses.

In this case, UBM with NSR=1/4 setting failed, and excessive queueing delays were observed.

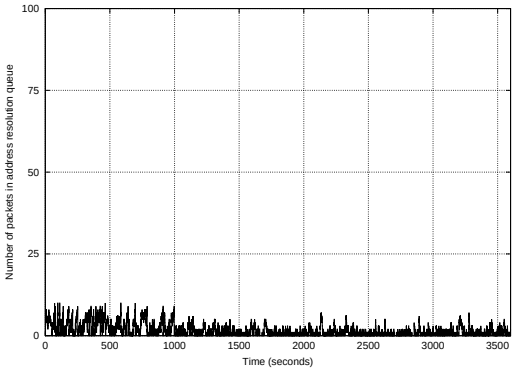
This suggests that after a crash UBM may be used in similar conditions as TCM, i.e. NSR=1. After the recovery period, the neighbor solicitation rate can be reduced to a lower value than TCM, e.g. NSR=1/4.



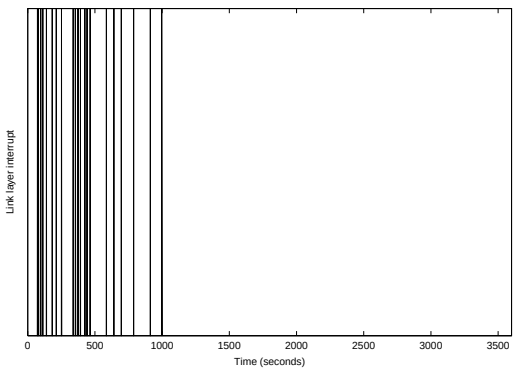
(a) Queueing delays.



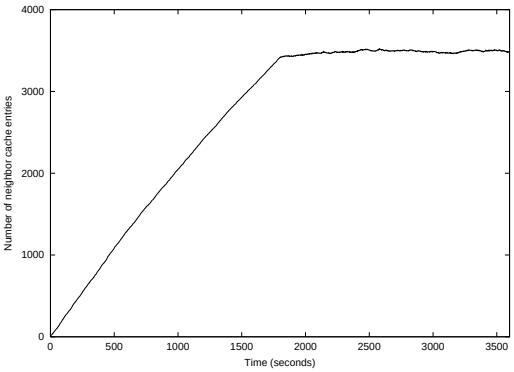
(b) Number of packets in address resolution queue (HBAR:OFF).



(c) Number of packets in address resolution queue (HBAR:TCM).

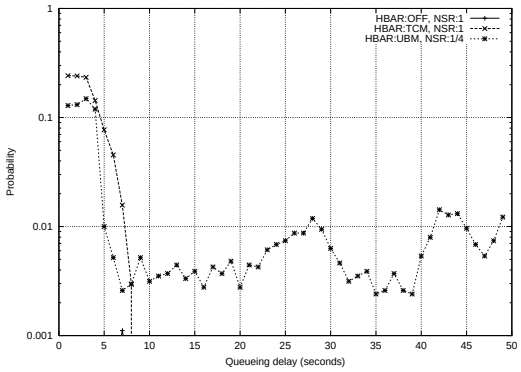


(d) Link layer interrupts (HBAR:TCM).

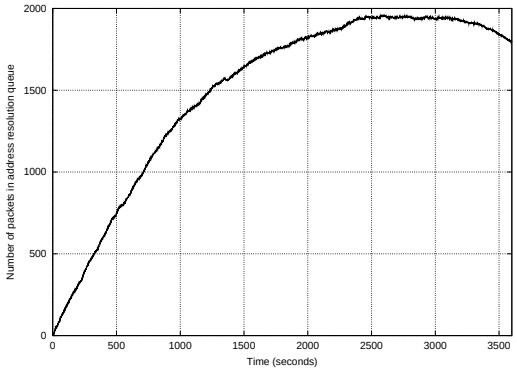


(e) Number of cache entries.

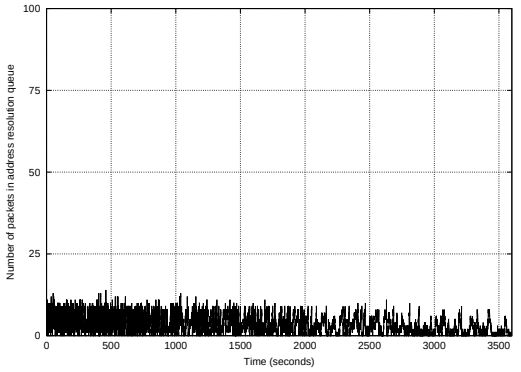
Figure 5.8: Subnet with 5,000 hosts.



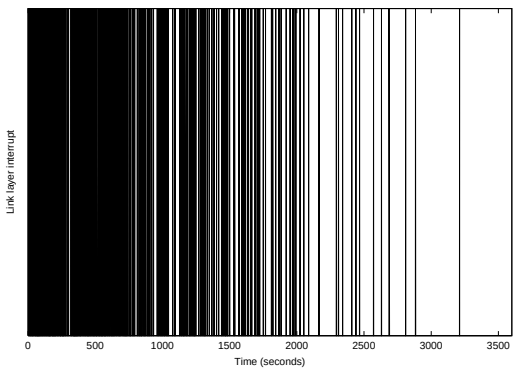
(a) Queueing delays.



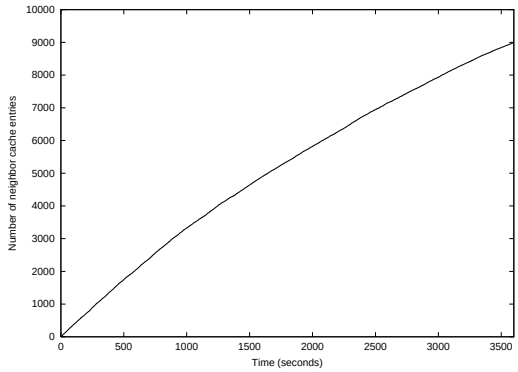
(b) Number of packets in address resolution queue (HBAR:OFF).



(c) Number of packets in address resolution queue (HBAR:TCM).



(d) Link layer interrupts (HBAR:TCM).



(e) Number of cache entries.

Figure 5.9: Subnet with 10,000 hosts.

Chapter 6

Discussion and conclusion

Contents

6.1 Summary

IPv6 Neighbor Discovery defines an address resolution mechanism to allow a node to dynamically learn and cache the link layer address of other nodes that are attached to it's local link. This procedure consists of multicasting a neighbor solicitation message over the subnet. A node that detects its IP address in the neighbor solicitation message, responds by sending its link layer address to the source. Address resolution is considered costly since it consists of multicasting a packet over the subnet. In large subnets with many hosts we expect a high incoming session rate which may result in high address resolution rate.

Thus, we have proposed a more conservative address resolution technique that that exploits the fact that IPv6 addresses addresses are long, i.e 128-bit. We call it Hashcast-based Address Resolution (HBAR). HBAR is less attractive for IPv4 with small (32-bit) addresses. HBAR is an advantage of IPv6 over IPv4.

We have shown by simulation that neighbor caches can effectively reduce the address resolution rate, provided that everything is running smoothly. In other cases, a faster mechanism than standard address resolution is desirable. There is a correlation between the incoming session rate and the required number of cache entries. Standard address resolution may suffer when the incoming session rate increases, until a sufficient number of addresses are resolved. Address resolution rate may also increase if for example an access router is low on memory and needs to prematurely garbage collect a large number of cache entries. In the worst case, an access router may crash and reboot with loss of state. The network may suffer from a high address resolution rate until a sufficient number of addresses are resolved. A mobile host may also change its link layer interface, in which case a large neighbor cache lifetime would lead to unreachability detection before the address resolved again. Thus, a larger session setup delay may be incurred by the correspondent node.

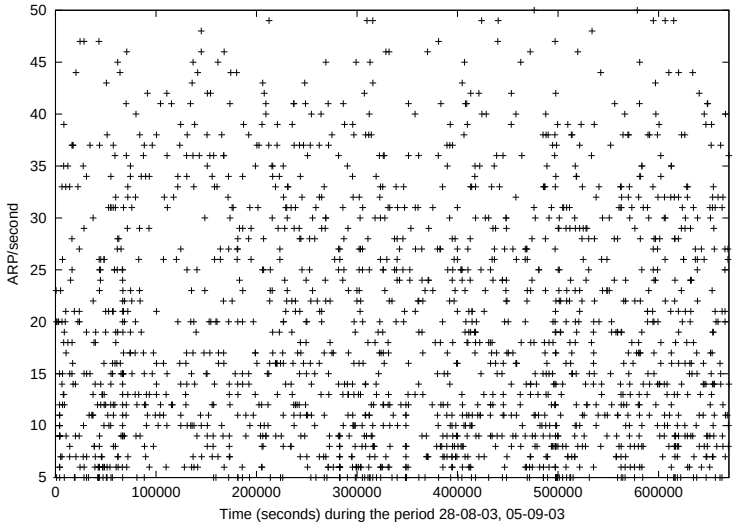


Figure 6.1: Address resolution traffic triggered by a virus.

6.2 Existing optimizations

We note that, concerning the address resolution cost, the design IPv6 Neighbor Discovery is already conservative. Unreachability detection is, in some sense, a technique for reducing multicast address resolution cost. The *STALE*, *DELAY* and *PROBE* states of the neighbor cache reduce the address resolution traffic. The reachability of a neighbor in *PROBE* state, is tested by sending unicast neighbor solicitations, hence multicast address resolution is avoided.

By current practice, in IPv4 networks, rate limited address resolution is generally recommended with for defending against viruses. Some viruses scan all addresses in range in order to discover new hosts to infect. These address scans generate broadcast ARP (Address Resolution Protocol) traffic. In order to control the bandwidth consumption, network administrators generally deploy rate limited ARP[20]. At time of writing, our network is infected by a virus, called LOVSAN (Just Wanted to Say Love You SAN). By coincidence we have been logging its duplication attempts and the resulting address resolution traffic in our network. Figure 6.2 shows the ARP rate (ARP/second) in our network during one week. We have many times observed up to 50 ARP/second, which confirms the rumors about viruses. However, our focus in this thesis is rather IPv6. Given the 2^{62} address possibilities per subnet, the usefulness of address scanning is questionable in IPv6. Therefore, currently we do not know if viruses or addresses scanners will increase the address resolution traffic in IPv6. We end this discussion without conclusion.

Some wireless access point vendors support a technique called *ARP filtering*, in order to reduce the bandwidth consumption of broadcasted ARP packets[21]. ARP filtering is useful in larges subnet with wired and mobile clients. For example, let 191.168.a.x the wireless IP address space and 191.168.b.x the wired address space. The access points in the network 191.168.a filter (i.e. drop) the ARP packets destined for hosts in 191.168.b.

6.3 Subnet size considerations

An important point to note is that in our simulations we have chosen relatively large subnet sizes. As a consequence, HBAR was necessary. There is an alternative solution. The total incoming session rate in a subnet can be very much reduced by configuring very small subnets. This is an alternative solution for avoiding address resolution delays.

On the other hand, it is hard to argue that small subnet sizes are better. The subnet size

depends on other factors than performance. A network administrator may wish to configure large subnets for efficient use of subnet prefixes. IPv6 supports up to 2^{62} hosts per subnet which is in practice an unlimited number. Note also that the management of access points may be considered easier and cheaper than managing IPv6 routers and routing tables. An access point acts as a wireless bridge, which allows easy and transparent network extensions. Thus, large subnets may be a more practical configuration alternative.

Large subnets also have performance benefits. In Mobile IPv6, a binding update signaling is generated by every mobile host and upon each handover, resulting in energy and bandwidth consumption. Large outdoor subnets may be preferable for reducing the Mobile IPv6 handover rate. Previous work addressed this problem using IP paging (that we will analyze in the next part of this thesis), however address resolution and large subnets can also reduce the binding update rate.

6.4 Link layer interrupts: How much do we care?

In HBAR, a neighbor solicitation is sent to the all-nodes multicast address. As a consequence, upon HBAR, every node in the subnet will receive an interrupt from its network interface card and process the neighbor solicitation packet (check the target address and drop). On energy constrained devices, frequent link layer interrupts may be considered harmful. In order to reduce the link layer interrupt rate we have defined the threshold controlled mode (TCM) of HBAR. In this case, HBAR is not used unless the number of packets in the address resolution queue reach a certain threshold, and thereby the link layer interrupt rate is reduced.

Confusingly, we note other sources of link layer interrupt. In IPv6 router advertisements are also send to all-nodes multicast address. The neighbor discovery protocol specification[8] limits routers to a minimum interval of 3 seconds between sending unsolicited multicast router advertisement messages. I.e. up to 1 link layer interrupt per 3 seconds is allowed. This rate is higher than most of the results that we have obtained with HBAR. More importantly, Mobile IPv6[22] specification (which will be reviewed in the next part of the thesis) recommends that router advertisement may be sent more frequently in order to help timely movement detection for mobile nodes; an optimization which is crucial for real-time applications e.g. VoIP. A mobile host detects its own movement by learning the presence of a new router as the mobile host moves into the router's wireless transmission range. A 50ms mean time between router advertisements is mentioned in the specification, in which case all hosts in the subnet will be interrupted 40 times per second on the average. This rate is very much higher than any link layer interrupt rate that we observed with HBAR.

6.5 The Neighbor Discovery DOS attack

HBAR has some relevance with a well-known denial-of-service (DOS) attack against IPv6 Neighbor Discovery.

In [23], Alper Yegin describes a possible DOS attack against IPv6 address resolution. An attacking node sends frequent packets to random interface identifiers at a target subnet. Given the 2^{62} possible suffixes, the destination addresses of the malicious packets are very likely to be non-existent and trigger address resolution. The impact that is mentioned in [23] is memory consumption due to the creation of a large number of neighbor cache states in $INCOMPLETE$ state. A solution is also proposed, in which the number of cache entries in $INCOMPLETE$ state is limited by the router.

We are rather worried about the bandwidth consumed in each cell of the target subnet, for multicast neighbor solicitations. From this perspective, HBAR improves the DOS-resistance by a factor of n (the number of concurrently resolved addresses), however complete DOS-resistance cannot be expected. Rather than resolving n addresses one-by-one, the access router can take n

queued addresses and resolve all of them concurrently. Hence, the address resolution queue that is filled by bogus addresses can be emptied quickly. There is no risk of neighbor advertisement storm, since the destination addresses are probably non-existent. If among n addresses there is a real address (being resolved for a real session), the legitimate address will be resolved and its packet will be delivered.

Unlike previous applications of HBAR, this application is best suitable for small subnets, since false neighbor advertisement rate will be smaller.

6.6 Multiple DAD at once

Duplicate Address Detection (DAD) uses the same mechanism as address resolution. In this section, we discuss the implications of our proposal on DAD and whether it is also useful in this context.

In DAD, upon generating a new (tentative) IPv6 address, a node checks if the tentative address is already in use by another node, by multicasting a neighbor solicitation message. If a node has indeed configured the same address, it replies with a neighbor advertisement, in which case address collision is detected and another address must be tried. Traditionally, DAD has been considered as a burden by the designers of FMIPv6 (Fast Mobile IPv6)[24], since it degrades handover performance of mobile hosts that in active IP communication with a correspondent node. Upon moving to a new subnet, a mobile host incurs a DAD delay until it has assurance that no other node has already configured its tentative IPv6 address.

A recent proposal, Optimistic DAD[25], says that the traditional approach it too pessimistic and the mobile host unnecessarily incurs DAD delay most of the time. This observation is based on the fact the interface identifiers in IPv6, are 62-bit long, hence the collision probability is very small. [26] states that in a subnet with 5,000 hosts the collision probability is smaller than 5.4×10^{-12} when interface identifiers are randomly generated. Thus in Optimistic DAD, upon entering a new subnet, a mobile host starts immediately to communicate using its tentative address rather than pessimistically waiting for a neighbor advertisement. If, however, the address is already owned by another node, the mobile host will suffer from disrupted communication. Note also that Optimistic DAD assumes that interface identifiers are randomly generated. Currently, we do not know if standard DAD is too pessimistic or Optimistic DAD is too optimistic.

In either case, using a Bloom filter, a node can detect multiple duplicate addresses at once, while the DAD delay is roughly the same as detecting one duplicate address. A node can multicast a neighbor solicitation message with a target address field (Bloom filter) that represents multiple tentative IPv6 addresses. A node can generate the tentative IPv6 addresses A and B, insert them into a Bloom filter and send a neighbor solicitation message (with the same format defined in Section 4.2). All nodes in the subnet can check their membership to the Bloom filter, if the address A already exist, the node A can detect it and reply with a standard neighbor advertisement. In this case the mobile host can directly use the address B. By performing multiple DAD, the node avoided another DAD delay for performing DAD on the address B. The node A may receive a false neighbor advertisement from a node C (which is not a member of the Bloom filter). However, C is not one of the tentative addresses of A, hence the false message can be silently dropped.

There is a possible DOS attack on DAD, which consists of spoofing positive responses to DAD attempts of a node, preventing it from configuring an IPv6 address[23]. When a Bloom filter is used, the tentative address(es) that is being tested is hided from the attacker. Given the 2^{62} address possibilities, the attacker cannot possibly find an address that is indeed represented by the Bloom filter. The only entity than can generate a valid neighbor advertisement is the node that has already configured the address being tested (in case of collision). The neighbor advertisement packet's source address will be the tested IPv6 address, proving that the sender knows i.e. configured the address. Note that, using a Bloom filter is not the only way of hiding the tentative address. A single one-way hash function H with 128-bit output would be enough. Let X the tentative address that is being tested. The querying node could send a neighbor solicitation

with the target address field set to $H(X)$. If X is already owned by another node, the node would detect it by comparing the target address field with $H(X)$ and return a neighbor advertisement message with X as the source IPv6 address, proving that it has configured X. Although this is a possible defense alternative, we promote the Bloom filter approach since it has performance benefits as well.

Part II

Optimizing the network-based IP paging services

Chapter 7

Review of Mobile IPv6 and IP paging

Contents

7.1 Mobile IPv6

Mobile IPv6 (MIPv6) allows IP nodes to remain reachable and to continue ongoing sessions while moving in the Internet[22]. Upon moving, i.e. changing its subnet, a node needs to configure a new IPv6 address. Without MIPv6, a moving IP node could not receive incoming sessions unless its DNS entry is also changed as proposed in [27]. Even if its DNS entry is changed upon each movement, an IP node would then not be able to maintain transport and higher-layer connections that need a fixed address during a connection. Frequently changing the IP address also makes difficult the maintenance of IPsec security associations that identify a end-node by its IP address.

Mobile IPv6 solves the IP mobility problem at the network layer by introducing a *home address*, *home network* and a new protocol called *binding update*. A home address is a MIPv6 node's fixed and publicly known IP address. A *home agent* is a new functional entity that is located in the home subnet and responsible for routing packets to a mobile node that is away from home. While way from home a mobile node configures a new IPv6 address in its visited subnet, called a *care-of-address*. The association between a node's home address and care-of-address is called a *binding*. A mobile node then sends a binding update message to its home agent and receives a binding acknowledgment message. The binding update message reports the mobile node's care-of-address, i.e. the IPv6 address at which the mobile node is currently reachable. The mobile node's binding is recorded by the home agent in a *binding cache*.

A *correspondent node*, an Internet node that wishes to communicate with a mobile node, transparently sends a packet to its publicly known and fixed home address. As a result, the packet is routed to the mobile node's home subnet where it is intercepted by its home agent. The home agent tunnels the packet to the mobile node's care-of-address found in the binding cache held. The correspondent node's packet is encapsulated in a another IPv6 header where the source and

destination address are the home agent's address and the mobile host's care-of-address respectively. The mobile node receives the correspondent nodes packet in its current subnet, decapsulates the received packet and finds the correspondent node's original packet sent to its fixed home address. At this point the communication between the mobile host and correspondent node can begin. There are two possibilities, bidirectional tunneling or routing optimization.

7.1.1 Bidirectional tunneling mode

Mobile IPv6 is designed to transparently work with any Internet node, i.e. even with correspondent nodes that are not aware of MIPv6. In this case, the mobile node and the correspondent node communicate using a mechanism called *bi-directional tunneling*. Packets from the correspondent node are routed to the home agent and then tunneled to the mobile node. Packets to the correspondent node are reverse tunneled from the mobile node to the home agent and then routed from the home network to the correspondent node. In this mode of operation, MIPv6 is completely transparent, the correspondent node is not aware that the other end of the communication is a MIPv6 host.

7.1.2 Routing optimization mode

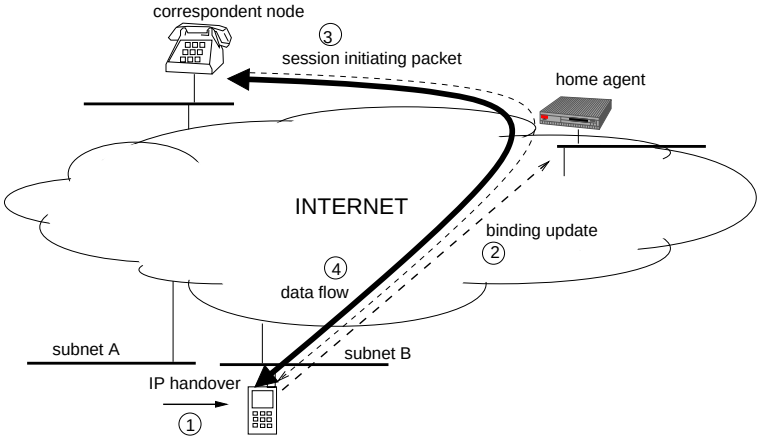
In the second of mode of operation, it is assumed that the correspondent node supports MIPv6. This mode is an optimization which is called *routing optimization*. The mobile host depends on its home agent only for receiving the first packet of an incoming session. Upon receipt of the correspondent node's packet (through its home agent), the mobile host sends a binding update message to the correspondent node which reports its current care-of-address. The mobile host and the correspondent node communicate without the routing service provided by the home agent, hence through a shortest path that directly connects both ends. The end-node addresses (inserted to destination and source IPv6 address fields) of the exchanged packets are the mobile host's care-of-address and the correspondent node's address. Each packet is also inserted the mobile host's home address, making the communication transparent to upper layers of the mobile host. Upper layer protocols at both ends, are given the impression that the end-point addresses are the mobile host's home address and the correspondent node's address. The correspondent node may be itself a mobile node, in which case the same procedure is used in both directions.

MIPv6 routing optimization has serious security implications. Mobile IP security will be reviewed in the next part of this thesis.

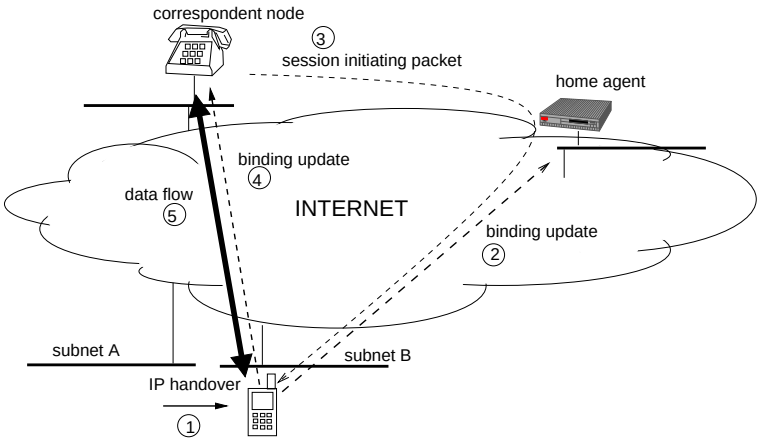
7.1.3 IP movements

A mobile host monitors periodic router advertisements to detect the presence of a new access router in range. In the absence of frequent router advertisements, or indications from the link layer, a host uses neighbor unreachability detection to detect when the previous router is no longer reachable. If a new router is discovered in the mean time, the mobile host configures a new care-of-address using the subnet prefix advertised by the new router and sends a binding update to its home agent and correspondent node (binding update to correspondent node is sent in routing optimization mode as described above).

If the mobile host is in active communication with a correspondent node, one or more packets en route from the correspondent node or home agent may be sent to the previous subnet, since the mobile host's binding update has not yet arrived at the destination. FMIPv6[24] solves this problem by letting the previous access router redirect the received packets to new care-of-address of the mobile host.



(a) MIPv6 bidirectional tunneling mode.



(b) MIPv6 routing optimization mode.

Figure 7.1:

7.1.4 Hierarchical MIPv6 (HMIPv6)

HMIPv6 introduces a new element called Mobility Anchor Point (MAP). In its domain (a set of subnets), a MAP acts as a topologically closer home agent to visiting mobile hosts that are away from home, in order to reduce the signaling cost and latency of binding updates. Upon entering the MAP domain, the visiting mobile host configures a *regional care-of-address* and sends a binding update to its home agent. The regional care-of-address points to the MAP's subnet. Upon moving in the MAP domain, the mobile host configures a care-of-address as usual. However in this case the mobile host does not need to send a binding update to its (topologically distant) home agent. Instead the mobile host sends a binding update to its MAP, updating the binding between its current care-of-address and regional care-of address, held by the MAP.

Upon incoming session, the correspondent node's first packet is tunneled by the home agent and then the MAP, and received by mobile host at its current care-of-address. At this point the mobile host sends a binding update to the correspondent node (if routing optimization is supported) with its regional care-of-address. Since the regional care-of-address is fixed, no other binding update to the correspondent node is required upon movement during the session. The correspondent node is given the impression that the mobile host is stationary and listens on the fixed regional care-of-address.

In summary, HMIPv6 reduces the binding update latency since binding updates are sent to a topologically closer MAP instead of a distant home agent. HMIPv6 also reduces the binding update signaling cost in the Internet since within MAP domain boundaries, the only binding update being sent to topologically distant home agents (and correspondent nodes) is the one that points to the fixed regional care-of-address.

7.2 Network-based IP paging service

IP paging is a research topic in progress for reducing the bandwidth and energy consumption in MIPv6 networks. The idea comes from existing cellular technologies, such as GSM[28], GPRS[29], CDMA[30]. These technologies allow the configuration of *paging areas* which cover many cells hence wide geographical coverage for idle terminals. Idle terminals do not update the network with their location unless they cross the paging area boundaries. Idle terminals are also allowed to enter an efficient dormant mode in which they listen to special paging channels and profit from time-slotted dormant mode by periodically turning on their receiver circuits. The idle terminals are not required to associate with their current base station.

Paging is an important optimization since terminals are idle most of the time. However, this service does not come for free. Upon call arrival a terminal must be paged, or searched in its most recent paging area, by broadcasting a paging message over every cell of the paging area. As the number of terminals per paging area grows, more frequent incoming calls arrive and more bandwidth is consumed for paging messages in each cell of a paging area.

IP paging aims to provide the same optimization using IP elements such as access routers and IP-addressable access points. IP paging has been proposed within the context of Mobile IP and IP micromobility protocols such as Cellular IP[31] and HAWAII[32][33].

In this chapter, we use the term IP paging to mean a network-based IP paging service. In Chapter 11 we will describe and discuss End-to-End IP paging, which is not a network-based service and has a different goal.

7.2.1 IP paging in Mobile IP

The first protocol (to the author's knowledge) that integrated IP paging into Mobile IPv4[34] was P-MIP: Paging Extensions to Mobile IP[35]. P-MIP assumes that all mobility related functions are handled by MIP and hence each cell is in a different IP subnet. This configuration leads to excessive binding update signaling. By configuring paging areas that cover multiple subnets, P-MIP reduces the binding update rate. Because paging areas provide wide geographical coverage

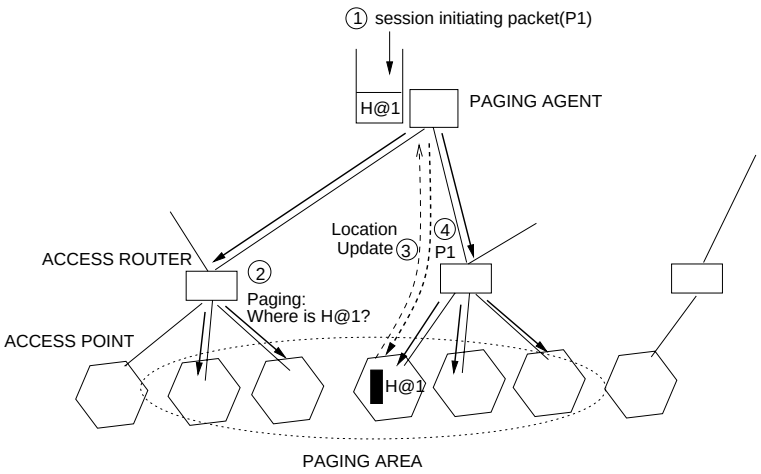


Figure 7.2: ORTH-IPP. IP paging using orthogonal subnets and paging areas, and paging agent. The mobile host is paged using its 128-bit home address found in the session initiating packet.

for idle hosts and mobile hosts are idle most of the time. Upon incoming session arrival, an idle mobile host's last registered access router (called foreign agent in MIPv4) initiates paging. Paging consists of broadcasting a paging message over every subnet of the paging area. The paging message contains the mobile host's fixed home address. The paged host detects the paging message, enters active mode and end-to-end communication begins.

The IETF analyzed the possible motivations of IP paging in MIPv6 networks and defined an IP paging architecture and a set of requirements [36][37]. The important component of the IETF IP paging architecture is a "paging agent". Paging agent is a new functional entity introduced to the Internet, and handles paging related functions. One of the IP paging requirements defined by the IETF is "orthogonality of subnets and paging areas", which means that paging areas and subnets may not perfectly overlap. A paging area may cover the cells of different subnets. This IP paging model is illustrated in Figure 7.2. An idle mobile host detects the address of the paging agent that serves its current paging area, and sends a binding update to its home agent. The binding points to the address of the paging agent. Upon incoming session, the first packet is routed by the home agent to the paging agent which buffers the packet and pages the destination host. The paging message contains the home address H@1 of the destination host. The idle host detects the paging message, sends a location update and receives the buffered packets. At this point, the mobile host acts as a regular MIPv6 host. I.e., enters active mode, configures a care-of-address in its current subnet and sends a regular binding update to its home agent. We call this IP paging model ORTH-IPP.

Another alternative is configuring subnets as paging areas and using Neighbor Discovery for IP paging, in which case the access router acts as a paging agent. IP subnets provide wide geographical coverage for active and idle hosts[38], idle hosts are paged using their care-of-address. This model of operation is illustrated in Figure 7.3, where C@1 is the destination idle host's care-of-address. In this case, the paging message will be the neighbor solicitation message, and a paged host will reply by sending a neighbor advertisement. We call this IP paging model ND-IPP. [1] It is important to note that in IP paging, an idle host does not associate with its current access point as it moves within the paging area (i.e. subnet in this case). Thus, neighbor caching is not possible. A session destined for an idle host must initiate paging (i.e. multicast neighbor solicitation). The problem in this case is that the access router must be able to differentiate between active and idle hosts.

[1] This might have better fit into the previous part of this thesis on neighbor discovery. However, previous work addressed the IP paging problem in its own context, and in theory ND-IPP and ORTH-IPP are the same service. Thus, we needed to discuss such use of neighbor discovery in the context of IP paging.

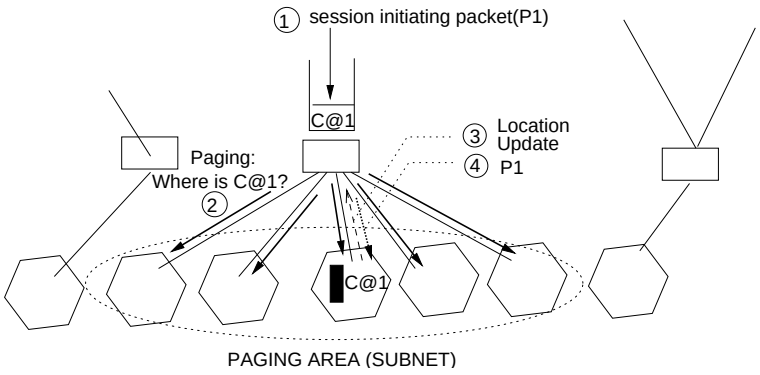


Figure 7.3: ND-IPP. IP paging using Neighbor Discovery (subnet=paging area). The mobile host is paged using its 128-bit care-of-address found in the session initiating packet.

In [38], we have proposed a stateful neighbor discovery mechanism. [2] Before entering idle mode, a mobile host registers with its current access router, and requests a paging service. Upon incoming packet destined for an idle host, the packet is buffered and paging is initiated, otherwise the packet is routed. This stateful approach, however, is not robust since the access router may reboot with loss of state. Other alternatives may be possible.

The advantage of ND-IPP, over ORTH-IPP is simplicity. The introduction of a new functional element to the Internet is avoided. However, flexibility is sacrificed to simplicity. Since subnets and paging areas are not orthogonal, the configuration of paging areas and subnets cannot be made separately, in case they differ in optimization and management criteria.

7.2.2 Adaptive IP paging

Adaptive IP paging is different from other IP paging protocols in that it also incorporates a paging optimization algorithm with the goal of minimizing the signaling costs of IP mobility[39]. Adaptive IP paging requires variable sized paging areas. A mobile node dynamically changes its paging areas size, according its mobility and incoming session arrival rate. Unlike the previous IP paging proposals that proposed static i.e. fixed sized paging areas designed for the aggregate, in adaptive IP paging the paging area sizes are adapted to individual host. This is motivated by the fact that each mobile host moves at a varying speed that is not necessarily same as other mobile hosts, and also has a varying incoming session rate that is probably different than other mobile hosts. Thus, the optimal paging area size is the one that is adapted to a mobile host's mobility and incoming session rate, rather than an average mobility and incoming session rate. For example, a mobile host that rarely receives incoming sessions can be given a large paging area than a mobile host that receives frequent incoming sessions. The former mobile host "merits" a large paging area since it introduces a smaller paging cost to the system. A larger paging area can further reduce the mobile host's battery drain and binding update rate. The latter host, that receives frequent incoming sessions, should be given a small paging area size since introduces a more important paging cost to the system.

While it is clear that adaptive IP paging is more optimal than static paging areas, it requires the configuration of variable sized paging areas, which is a challenging task. In Chapter 9 we will describe one possible architecture and paging area configuration algorithm to address this problem.

[2] Alper Yegin proposed it before us (unfortunately we do note have any reference to cite), but for defending against the neighbor discovery DOS attack that we have reviewed in Section 6.5.

7.2.3 Review of time-slotted paging

In current cellular systems terminals are allowed to enter an efficient power save mode which is generally called *time-slotted dormant mode*[28]. A time period called *slot cycle* is divided into several time slots. A dormant terminal's receiver is synchronized with the network and during a slot cycle, the terminal listens only one slot. The rest of the time, its receiver circuit is off. The problem is that, upon call arrival, the paging message must by broadcasted in the paged terminal's time slot. Upon link layer movement, the terminal does not generate any signaling, hence its time-slot is ignored by the system. In current systems, this problem is solved by choosing the time slot as a function of the terminal identifier, e.g. the 2 or 3 least significant bits of the terminal identifier. For example, with 2 bits, $2^2 = 4$ different time slots are obtained. Upon incoming call, the access points find the paged terminal's identifier in the paging message and hence its time-slot.

Chapter 8

Hash-based IPv6 paging

Contents

8.1 Introduction

In the previous chapter we have reviewed two possible IP paging implementations, ORTH-IPP and ND-IPP. These alternatives differ in network management flexibility and implementation complexity aspects. However, from theoretical viewpoint ORTH-IPP and ND-IPP offer the same basic service and optimization which is IP paging. In the first one, subnets and paging areas are orthogonal, a paging agent is used and a host is paged using its home address. In the second approach, subnets are configured as paging areas, Neighbor Discovery is used, and a host is paged by its access router using its care-of-address.

In this chapter we describe a hash-based IP paging protocol using Bloom filters. In ND-IPP, Bloom filters are applied to care-of-addresses, whereas in ORTH-IPP there are applied to home addresses. This is illustrated in Figures 8.1-8.2. Note that care-of-address and home address are 128-bit long, thus offer the same hash coding performance.

8.2 Mobile host states

- Active: A mobile host that actively communicates with a correspondent node. The exact location of an active mobile host is known to the system.

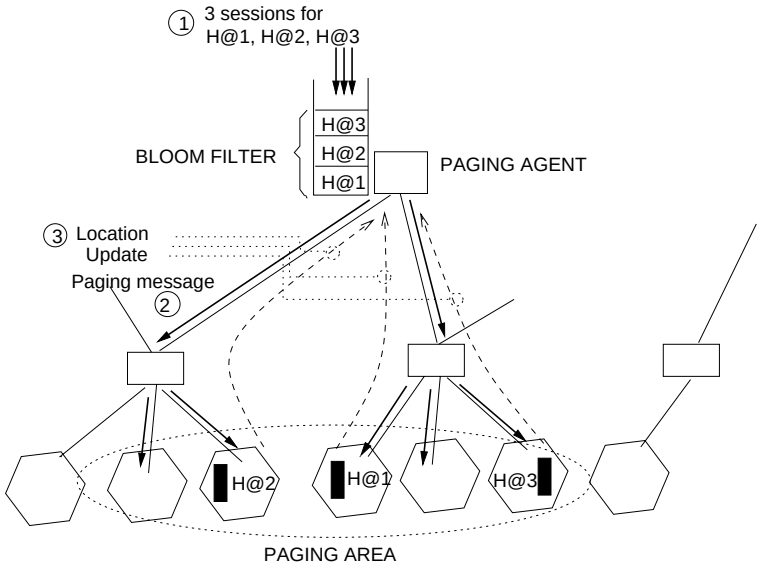


Figure 8.1: Hash-based ORTH-IPP using 128-bit home addresses.

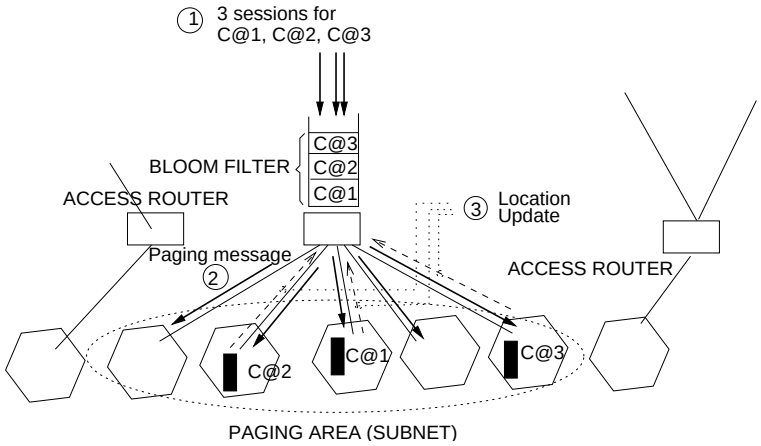


Figure 8.2: Hash-based ND-IPP using 128-bit care-of-addresses.

- Idle: A mobile host that is not in active communication enters idle mode. An idle host may move within paging area boundaries without reporting its location. Therefore, upon incoming session it must be paged.
- Dormant: Same as idle state, except that the host also profits from a time-slotted dormant mode for reducing the energy consumed by its receiver circuit.

8.3 Hash-based IP paging

8.3.1 Protocol description

In this section, we develop a hash-based IP paging algorithm that can possibly be used by any IP paging implementation. The addresses of several idle hosts are represented by a Bloom filter, the paging message is inserted the Bloom filter and broadcasted. Multiple hosts are paged concurrently using a single paging message, and thereby bandwidth is saved in each cell[40][41].

n	k_{opt}	F	$F_{C=4}$
2	44	0.000000	0.000000
3	29	0.000000	0.000000
4	22	0.000000	0.000000
5	17	0.000005	0.000001
6	14	0.000035	0.000009
7	12	0.000153	0.000038
8	11	0.000459	0.000115
9	9	0.001078	0.000269
10	8	0.002134	0.000533
11	8	0.003732	0.000933
12	7	0.005947	0.001487
13	6	0.008821	0.002205
14	6	0.012367	0.003092
15	5	0.016575	0.004144

Table 8.1: The number of hash functions k that minimizes the false location update probability F. The k and F values are set according to well-known equations Equ.3.1 and Equ.3.2 with $m = 128$.

The procedure requires k_{max} uniform independent hash functions $h_1(), h_2(), ..., h_{k_{max}}()$, where $1 \leq h() \leq 128$, as previously described. Let $A = \{IP_1, IP_2, ..., IP_n\}$ a set of n idle host IP addresses waiting in the paging queue (home address or care-of-address depending on IP paging implementation). For each element IP_i in the set A, $h_1(IP_i), h_2(IP_i), ..., h_k(IP_i)$ is calculated and the corresponding bit positions of a 128-bit Bloom filter is set. The resulting Bloom filter and k are inserted to the paging message, which is broadcasted over the paging area. Upon receipt of the paging message, an idle host IP_j in the paging area, checks the previously stored bit positions $h_1(IP_j), h_2(IP_j), ..., h_k(IP_j)$ of the Bloom filter in the received paging message. If all of them are 1, then the mobile host detects that it is being paged, and sends a location update.

A false positive will lead to what we call a "false location update", i.e. the mobile host will unnecessarily wake up send a location update. We assume that k is optimized in order to minimize the false location update probability. The false positive probability F and optimal number of hash functions k_{opt} are shown in Table 8.1.

8.3.2 Time-slotted dormant mode support

In IP paging, the time-slot of a paged host must be determined by the access points regarding the IP address of the paged host. The hash-based paging procedure that we have defined in the previous section, hides the identifier of a paged host from access points. As a consequence access points cannot determine the time-slots of paged hosts.

In order to support time-slotted dormant mode, Bloom filters can be applied by access points. This procedure is illustrated in Figure 8.3. In this example, the time-slots are determined using the 2 least significant bits of paged host IP addresses, hence we have $2^2 = 4$ different time-slots denoted ts_1, ts_2, ts_3, ts_4. The access point receives 3 broadcasted paging messages (so the other access points in the paging area). Two messages page two hosts with IP addresses that correspond to ts_4. These hosts can be concurrently paged using a Bloom filter. The access point prepares a Bloom filter using their IP addresses, and broadcasts one paging message instead of broadcasting two paging messages in its cell.

Note that with time-slotted dormant mode, the false location update probability is reduced by a factor of C, where C is the number of time-slots. Because a dormant host hears only $1/C$ of the broadcasted paging messages. The false location update probabilities with 4 time-slots denoted $F_{C=4}$ are shown in Table 8.1.

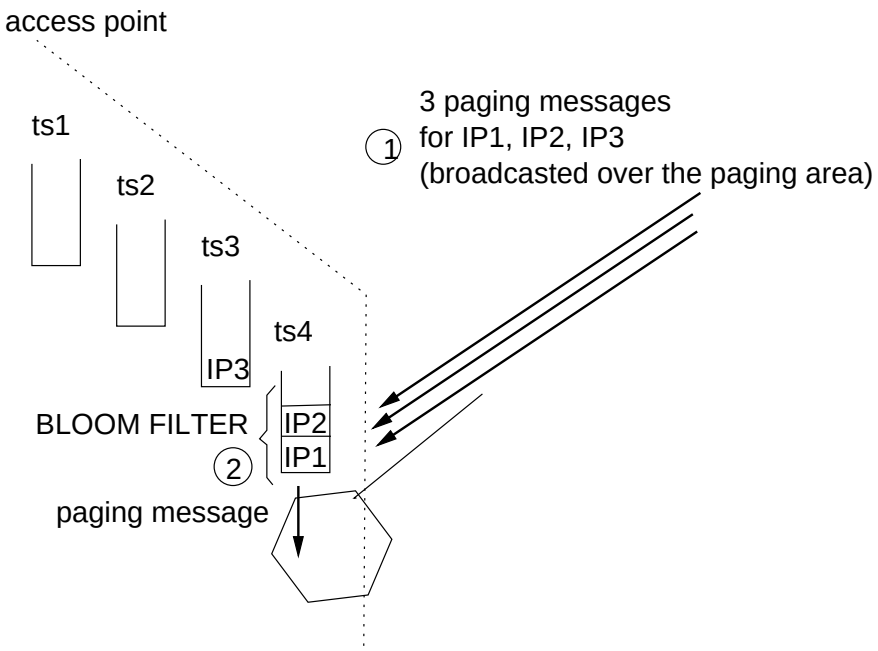


Figure 8.3: Bloom filter applied by an access point.

8.3.3 Link layer interrupts

The details of IP paging are not yet well defined. We assume that link layer multicasting is always available and mobile hosts are paged using a mechanism that is similar to solicited-node multicast address, in order to minimize the link layer interrupt rate on mobile hosts. As a consequence, we will assume that our proposal has a drawback that standard IP paging has not. Idle mode hosts' IP modules will receive link layer interrupts upon each hash-based IP paging event in their paging area.

On the other hand, without justification, we will assume that link layer interrupts are not harmful when time-slotted dormant mode is available, because the dormant hosts will not hear the paging messages that are broadcasted over their paging area, most of the time.

8.3.4 Bandwidth gain

In hash-based IP paging, 1 paging message is broadcasted over the paging area instead of n, at false location update cost. Assuming equal sized paging and location update messages, the bandwidth gain is:

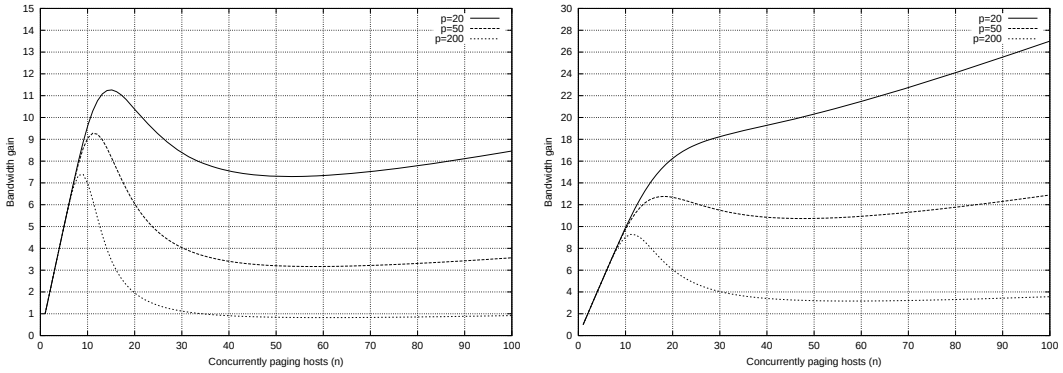
$$G_{core} = \frac{n}{1 + N \times F} \tag{8.1}$$

$$G_{wireless} = \frac{n}{1 + p \times F} \tag{8.2}$$

where N and p are the number of idle hosts per paging area and cell, respectively. Figure 8.4-a, shows the bandwidth gain without dormant mode support.

With time-slotted dormant mode, Bloom filters will be applied by access points and F should be replaced by F/C. In this case, access points can filter the false location updates, therefore we have:

$$G_{core} = 1 \tag{8.3}$$



(a) Without dormant mode support

(b) With dormant mode support and 4 time-slots.

Figure 8.4: Bandwidth gain per cell.

$$G_{wireless} = \frac{n}{1 + p \times F/C} \tag{8.4}$$

The wireless bandwidth gain with dormant support (and 4 time slots) is shown in Figure 8.4-b.

8.4 Modes of operation

When dormant mode support is not available, we will prefer hash-based IP paging in threshold controlled mode (TCM) in order to reduce link layer interrupt rate. In this case, hash-based IP paging will be performed whenever

$$n \geq 9$$

and 9 hosts will be concurrently paged. This setting guarantees that an idle mode host will not be disturbed by false location updates more than once per 1000 paging events in its paging area, for any configuration (recall that false positive probability with $n = 9$ is $F \simeq \frac{1}{1000}$). In this case, for example, in a paging area with $N = 4,000$ idle hosts we expect a bandwidth gain of $G_{core} = 1.69$ each time hash-based paging is performed. On the other hand in a paging area with $N = 8,000$ idle hosts, hash-based paging will result in small bandwidth loss and $G_{core} = 0.94$. The wireless bandwidth gain depends on the cell size. Assuming macro cells with $p = 200$ we expect $G_{wireless} = 7.4$. Assuming micro cells with $p = 20$ we expect $G_{wireless} = 8.54$.

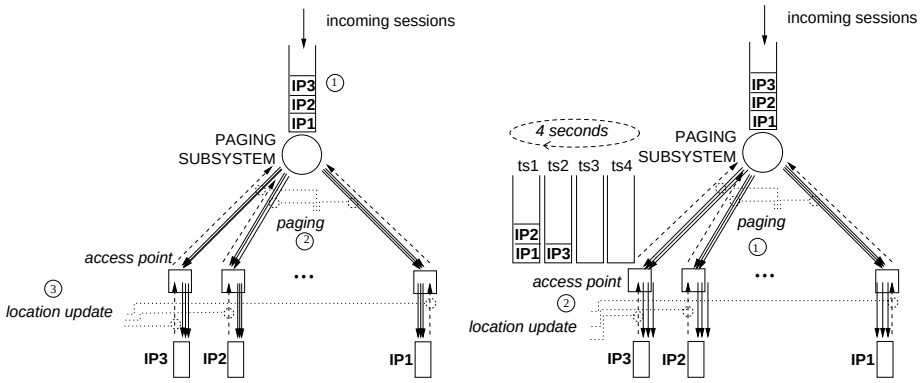
With dormant mode support, we will prefer hash-based IP paging in upper bounded mode (UBM). Hash-based IP paging will be performed whenever

$$n > 1$$

and up to 9 hosts will be concurrently paged. A higher upper bound than 9 can also be chosen since the false location update probability is smaller with time-slotted dormant mode.

8.5 Performance analysis

In this section we evaluate the performance of standard IP paging and hash-based IP paging. In either case large paging areas are required in order to reduce the MIPv6 binding update rate in the core network, and reduce power drain on mobile hosts. However, large paging areas also increase the paging message rate in every cell.



(a) Standard IP paging.

(b) Standard IP paging (time-slotted dormant mode).

(c) Hash-based IP paging.

(d) Hash-based IP paging (time-slotted dormant mode).

Figure 8.5: Simulation environment.

We limit the IP paging rate in order to exploit the bandwidth gain of hash-based paging. Therefore, as the paging area sizes grow, we expect session setup delays due to queueing delays in the paging sub-system. We measure the queueing delays with standard IP paging and hash-based IP paging.

8.5.1 Simulation overview

Our simulation environment looks like Figure 8.5. Standard IP paging and hash-based IP paging are applied to the same network topology, i.e. a paging area that covers a number of access points. The paging sub-system (paging agent or access router depending on how IP paging is implemented) deploys a paging queue. In standard IP paging, one idle host is paged at a time. In hash-based IP paging, several idle hosts in the paging queue can be paged at the same time and bandwidth can be saved.

When time-slotted dormant mode is used, the paging messages are buffered by access points. In this case, each access point deploys a different paging queue that corresponds to each time slot. Bloom filters (when turned on) are applied by access points. We assume that time slot cycle duration is 4 seconds and 4 time slots are deployed.

ips	$I = 1$	$I = 2$	$I = 3$
$N = 1,000$	0.290	0.568	0.804
$N = 2,000$	0.573	0.931	0.973
$N = 3,000$	0.804	0.973	0.989
$N = 4,000$	0.921	0.985	0.996

Table 8.2: Link layer interrupt rate with hash-based IP paging.

In each cell, the paging rate is limited to 1 paging message per second. We simulate one hour of incoming sessions using different paging area sizes. Paging area size N is reported in terms of the number of idle mobile hosts per paging area. Incoming sessions are driven by a Poisson process, and the average incoming session rate per hour of a mobile host is set to $I = 1, I = 2$ and $I = 3$ in different simulations.

The following measurements are made during one hour of simulation:

- Queueing delays in the paging sub-system (in seconds).
- *ips* (interrupt per second). The average rate of link layer interrupts (per mobile host).

8.5.2 Queueing delays without time-slotted dormant mode

When dormant mode support is not available, hash-based IP paging is used in TCM mode with the goal of reducing the link layer interrupt rate on mobile hosts.

We limit the IP paging rate to 1 paging/second, in order to profit from the bandwidth gain of hash-based IP paging. Figures 8.6 and 8.7 show the queueing delays with standard IP paging and hash-based IP paging algorithms. Standard IP paging failed with increased I or N, since paging messages are rate limited. With hash-based IP paging all idle hosts were located within reasonable delays.

Table 8.2 gives the link layer interrupt rate *ips* caused by hash-based IP paging. The TCM mode along with the rate limited paging policy kept the link layer interrupt rate under 1 interrupt per second. Clearly, as the paging area size and/or incoming session rate increases the TCM threshold is reached more frequently, in which case hash-based IP paging is performed more frequently and the link layer interrupt rate increases.

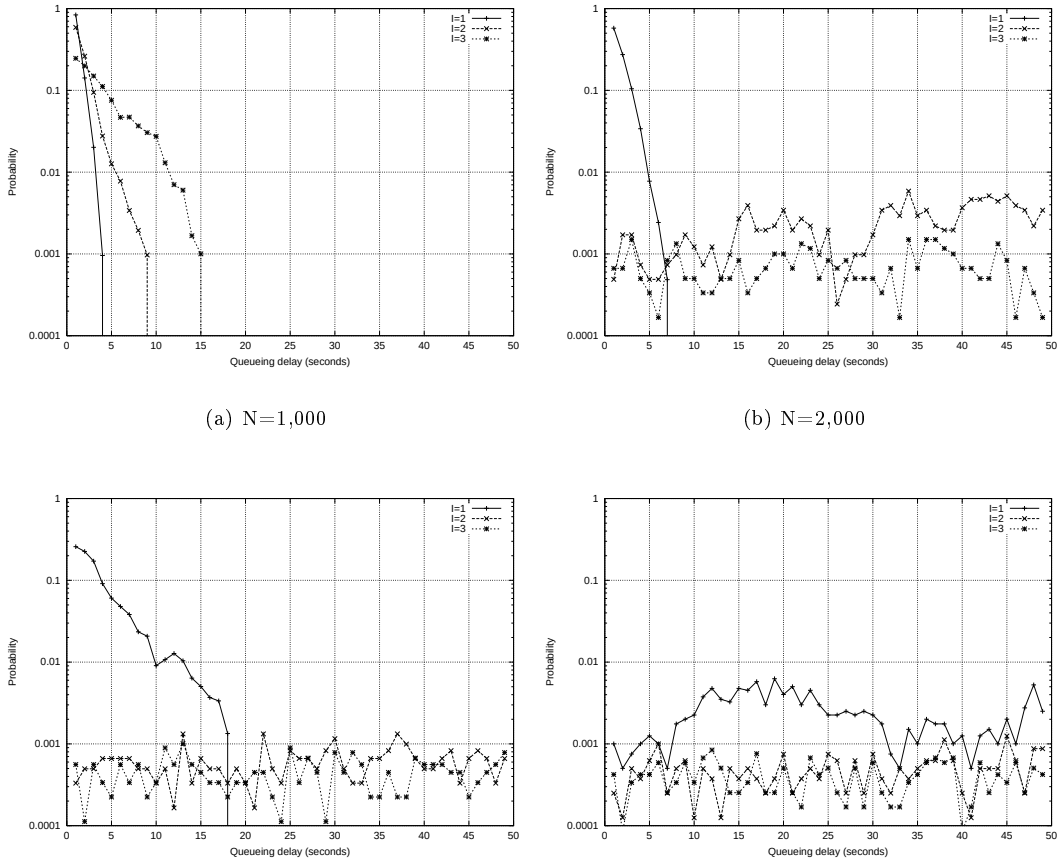
8.5.3 Queueing delays with dormant mode support

In the following simulations, time-slotted dormant mode support is available. Time-slotted dormant mode reduces the battery drain on dormant hosts, at the cost of increased paging delays. We simulate a 4 second slot cycle with 4 time-slots. The total paging message rate in each cell is 1 paging message per second. However, the paging rate per time-slot is 1/4 paging per second. As a consequence, we expect longer paging delays than the previous simulation, although the bandwidth that we reserve for paging is the same.

With time-slotted dormant mode support, hash-based paging is used in UBM mode. In this case, we expect a higher rate of hash-based paging, but $ips \leq 1/4$ since a dormant host's receiver is in off position 3/4 of time.

Each access point deploys 4 queues, one for each time-slot. Paging messages are received from the paging sub-system and accumulated in 1 of 4 queues depending on the dormant host identifiers found in the paging message. With standard paging, the access point can serve 1 paging message during a time-slot (the next message must wait 4 seconds before being broadcasted). With hash-based paging, the access point can serve up to 9 paging messages during a time-slot.

The paging delays are shown in Figures 8.8 and 8.9. Standard IP paging could not cope with most of our settings, excessive queueing delays occured. Hash-based IP paging in UBM mode



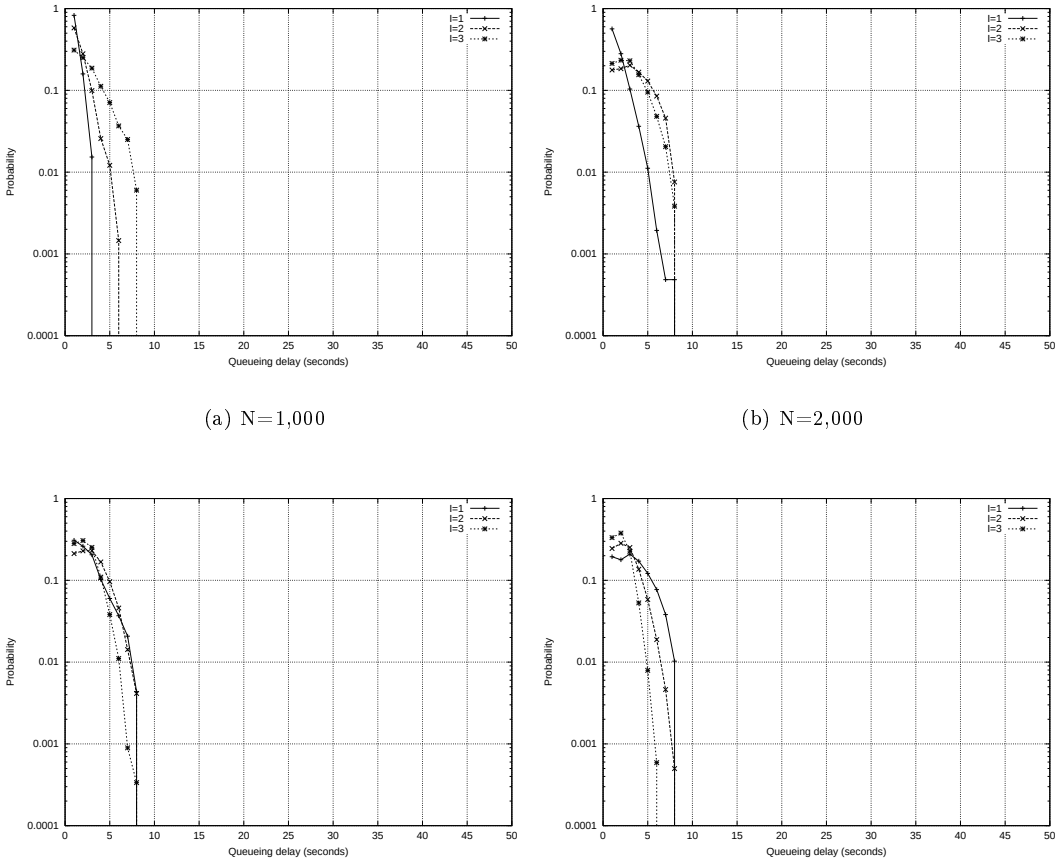
(a) N=1,000

(b) N=2,000

(c) N=3,000

(d) N=4,000

Figure 8.6: Queueing delays with standard IP paging.



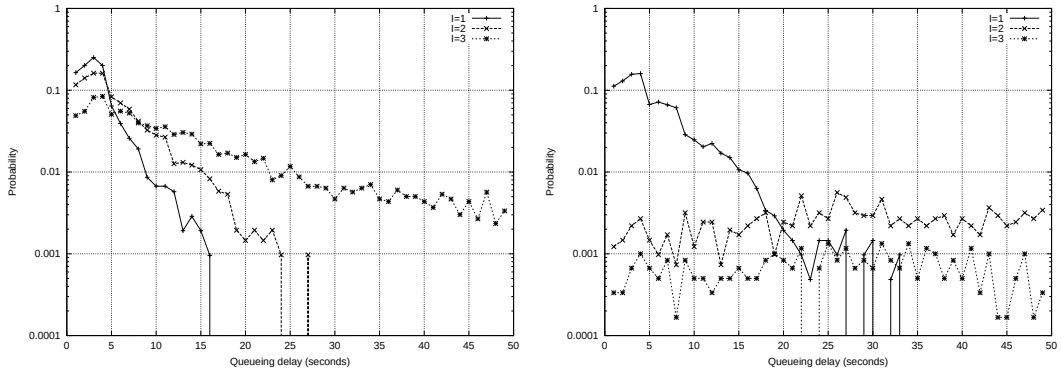
(a) N=1,000

(b) N=2,000

(c) N=3,000

(d) N=4,000

Figure 8.7: Queueing delays with hash-based IP paging.



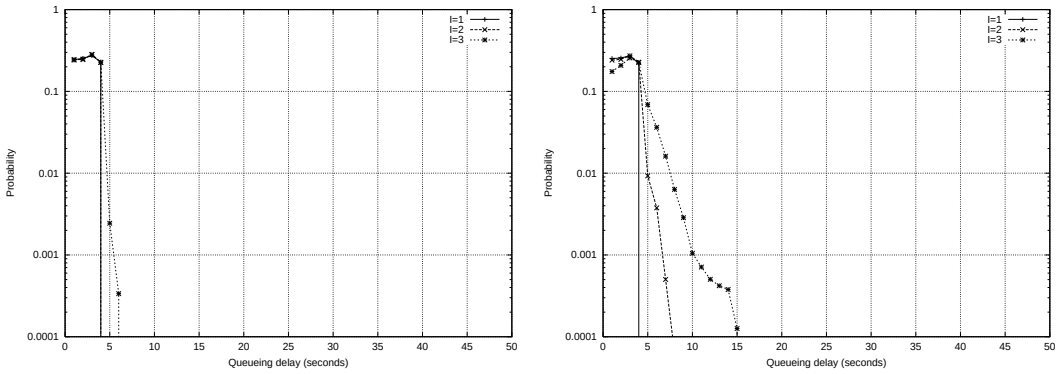
(a) N=1,000 (b) N=2,000

(c) N=3,000 (d) N=4,000

Figure 8.8: Queueing delays with standard IP paging (time-slotted dormant mode).

outperformed the standard paging with $N = 4,000$ also a larger paging area size with $N = 8,000$ dormant mode hosts. With $N = 8,000$ and $I = 3$ settings a small number of session suffered from a queuing delay that is greater than 10 seconds.

The link layer interrupt cost is reported in Table 8.3. Since the receiver circuit of a mobile host is off $3/4$ of times, the link layer interrupt rate cannot be higher than $1/4$. When N and I are small the link layer interrupt rate is smaller since hash-based IP paging is not performed when $n = 1$, i.e. standard IP paging is performed (note however that we "assume" a standard IP paging protocol that employs a technique similar to the solicited-node multicast approach of IPv6 address resolution). As we increase N and I, we have $n > 1$ most of the time in each time-slot and the link layer interrupt rate approaches to $1/4$.



(a) N=4,000 (b) N=8,000

Figure 8.9: Queueing delays with hash-based IP paging (time-slotted dormant mode).

ips	$I = 1$	$I = 2$	$I = 3$
$N = 4,000$	0.169	0.221	0.238
$N = 8,000$	0.214	0.245	0.249

Table 8.3: Link layer interrupt rate (time-slotted dormant mode).

Chapter 9

An adaptive IP paging architecture

Contents

9.1 Introduction

Traditional cellular systems, such as GSM [28], employ *static* paging areas. An operator manually configures the paging areas (i.e. their sizes and shapes) according to the average mobility and traffic characteristics in a given geographical area. All hosts use the same paging areas. Previous IP paging proposals adopted the same approach, i.e. they defined static paging areas. Adaptive IP paging improves the paging performance by optimizing the paging area sizes[39] as we have previously reviewed.

However, adaptive IP paging requires the configuration of variable sized paging areas. In this chapter, we develop an architecture and algorithm for automatically configuring variable sized and optimal shaped paging areas[42][43].

9.2 Architecture

9.2.1 Paging area model

Adaptive IP paging requires orthogonal paging areas and subnets, i.e. the model of operation is the ORTH-IPP model that is previously described. In this chapter, we assume that subnet and cell boundaries are identical. I.e. each cell is in its own subnet and identified by a subnet prefix.

In the proposed architecture, a paging area is defined as a list of subnet prefixes which form the paging area. Paging area boundaries are detected by comparing the subnet prefix advertised by the current access router against the list of subnet prefixes. If the current subnet prefix does not match, then the host has left its paging area. Whenever an idle host moves out of its current paging area, it "downloads" a new one from its paging agent. Figure 19.6 illustrates an adaptive paging system. In this figure a mobile host enters idle mode in cell c_1. Before entering idle mode the host requests (from its paging agent) a paging area that is relative to cell c_1 denoted pa_1. The shape and size of pa_1 are adapted to the host's mobility and communication characteristic. The host then moves out of pa_1 at cell c_2. At this point it requests from its paging agent a new optimized paging area pa_2 and so on.

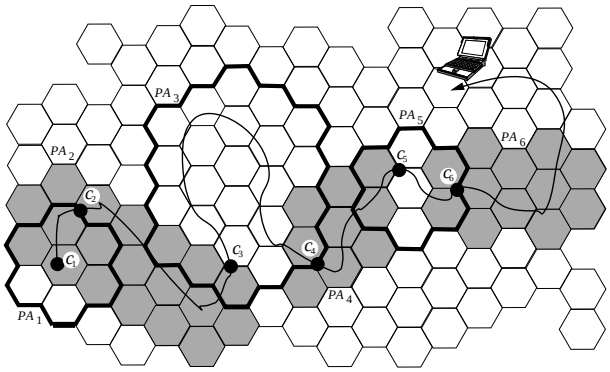


Figure 9.1: Adaptive paging area sizes and shapes (solid dots represent binding updates).

9.2.2 Network aggregated paging area shapes

In the proposed architecture each host uses time-varying and individual sized paging areas. The paging area sizes and shapes are dynamically adjusted. As described in [39], the optimal size of a paging area essentially depends on the host's communication pattern (i.e. incoming data session rate) and its speed (i.e. how often it moves from one cell to another).

On the other hand, the optimal shape of a paging area depends on host movement direction probabilities. For example, along a highway a host would better benefit from a linear paging area. In contrast, a host moving in an urban area with a random pattern would benefit from a symmetrical, i.e. a circular paging area.

There are two possible configuration alternatives for obtaining optimal paging area shapes. We choose the second one:

- Each host configures fully personal paging areas which adapt well to its mobility characteristics. The host records a mobility history which gives information about the most probably visited cells. This information can be used for deriving optimal paging area shapes relative to each cell. However this solution has two major drawbacks: (1) the memory requirement on hosts can be excessive (2) the mobility history might not be always available (i.e. when the host visits a site for the first time) or out-of date (for example when access points are added or removed). More importantly, there are public areas which are exposed to high rates of visits although many of the users rarely (if more than once) visits those areas. Good examples are highways and railways on rural areas. Although most of the users travel very rarely, we always observe important user flow rates in such places. Intuitively we can say that in those places, paging area configuration based on personal mobility patterns may result in high rates of binding updates since hosts will not have enough paging area information.

- The network derives "aggregated" paging area shapes based on the information sent by every host in its coverage. The resulting paging area shapes might not be always optimal for each

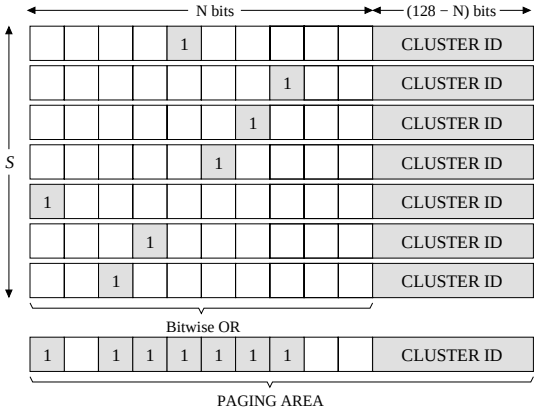


Figure 9.2: Paging area coding.

individual host (i.e. each host will use the same paging area shapes). However this is a "low-cost" and "scalable" solution. Paging area configuration effort on hosts is minimized, and paging areas can rapidly converge. A host can obtain paging areas even in the domains that it visits for the first time. The paging area shapes will adapt to local movement characteristics in that domain which will be more or less similar for each host. We have therefore adopt this paging area auto-configuration policy.

To summarize, in our architecture, *the paging area size is personal and computed by the mobile host, whereas the paging area shape is aggregated and computed by the network.*

Given these design choices, we face the following problems:

1. *How does a host compute its optimal paging area size?* We do not propose any new solution to this problem. We make use of the algorithm that was developed in [39].

2. *How does the network configure optimal shaped and variable sized paging areas?* A low-cost and scalable solution to this problem is proposed in section 9.3.

9.2.3 Paging area coding

Recall that in the proposed architecture a paging area is represented as a list of subnet prefixes and "downloaded" by mobile hosts. When paging area sizes are large, the transmission of this information over wireless links may consume a valuable portion of precious bandwidth and increase the packet error probability. Furthermore, the reception of large packets may represent an energy consumption overhead on hosts. As a result, it is important to have a scheme which allows for compressing paging area information.

[1] We propose a bitmapping procedure based on the cell addressing scheme illustrated in Figure 9.2. The network is divided into clusters. A cell is coded using 128 bits where the 128-N least significant bits represent a binary cluster ID. The remaining N bits identify a cell within a cluster. Each cluster contains N cells therefore only one bit is needed to identify a cell in a given cluster. Each access router broadcasts this information in its router advertisement messages. This is manually configured by the operator. Then, a number of cells (S in Figure 9.2) which belong to a same cluster can be coded as a paging area represented by a single 128 bit identifier: a cluster ID and the bitwise OR of the bits corresponding to the cells included in the paging area.

Upon movement, a mobile host can check if it has crossed the boundaries of a given paging area, as follows:

[1] This algorithm is designed by Claude Castelluccia.

if $\neg(c \wedge pa)$
then send binding update

where c and pa are the current cell and paging area of the mobile host.

Different parts of a paging area may fall in different clusters. Clearly, in this case the coding performance will be reduced since more than one 128 bit slots will be needed to represent the paging area. However, N can be chosen large in order to reduce the probability of such a situation. Note that since this addressing scheme is not used for routing, the cluster IDs are reusable. However, any two clusters having the same ID, should be enough distant from each other in order to avoid any confusing paging area information. This criterion decides the upper bound on N.

9.3 Paging area configuration algorithm

In the proposed architecture, the cellular network is divided into domains. There is one Paging Area Configuration Agent (PACA) per domain and each PACA is responsible for configuring variable sized and optimal shaped paging areas relative to each cell in its domain.

A PACA uses a paging area auto-configuration algorithm that consists of two major parts: *sampling* and *paging area composition*. Sampling is the process of extracting the direction probabilities in each cell (i.e., the probability of moving to a given neighboring cell). Composition is the process of configuring a paging area relative to a given cell.

9.3.1 Sampling

Sampling is the most challenging part of paging area auto-configuration. A sampling algorithm should be low-cost and scalable. By low-cost, we mean that the algorithm should not call for excessive CPU operations which will degrade the performance of hosts and/or the network.

A sample is an ordered pair of adjacent cells $[a, b]$. Samples are generated randomly by mobile hosts. When a mobile host crosses the boundaries of its paging area, it sends a binding update to its home agent and registers with its next paging agent (as defined by previous IP paging proposals). In our case, however, the mobile host also sends a sample to its PACA with a very small probability (e.g. p=1%). The probability of sending a sample is low so that sampling has no power consumption overhead on mobile hosts and negligible bandwidth cost (1% of the binding update cost). The identity of the host which sends a given sample is not taken into account. As a consequence, the algorithm captures the aggregated movement characteristics which are more or less common to each individual host. This sampling policy is at the heart of paging area configuration scalability and illustrated in Figure 9.3.

In a cellular Internet with hundreds of millions of hosts, the number of samples will not be problem even with negligible sampling rates. This issue will be further discussed in Section 9.4. Secondly, each PACA is responsible for sampling and paging area configuration in its own domain. As a result, the paging area configuration effort is distributed.

Sampling is a temporary process and continues until the paging areas relative to the cells in the PACA's domain converge. Once the paging areas converge, the PACA distributes the paging areas to the relevant paging agents in its domain. The details of this process is beyond the scope of our work. Later the PACA can restart sampling in order to adapt some new changes in the cellular or geographical topology. This can be automatic or manually initiated by the operator.

9.3.2 Paging area composition

The collected samples are processed by the PACA in order to extract the direction probabilities in each cell in its domain. Then, the PACA composes the paging areas. These processes run periodically when the CPU load is low and until convergence (controlled by a daemon process for

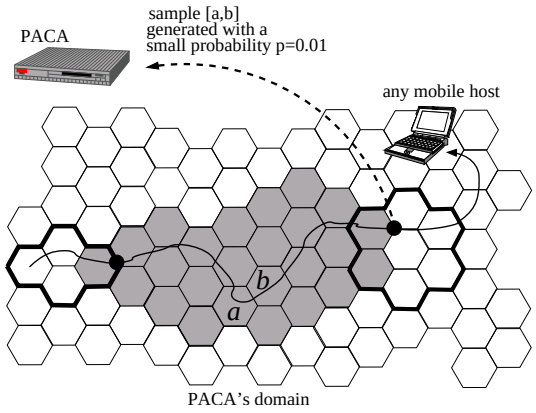


Figure 9.3: Sampling procedure.

example). If these processes run at night, the following day, paging areas will be more efficient. Clearly, this will increase the convergence time of paging areas. However, paging area configuration is a temporary process and convergence delays up to several days is acceptable. This allows very low rates of processing.

The PACA uses the procedure **compose**(x) shown in Figure 9.5 for composing a paging area relative to a given cell x denoted pa_x. The composition procedure begins with an empty pa each time it is executed. This is necessary for adapting to most up to date movement direction probabilities. The first cell which is added to the pa_x is x itself (lines 0-1). $P_x(i)$ is the probability of visiting the cell i without leaving pa_x and we have always $P_x(x) = 1$. Next, we find the neighboring cells of the pa using the movement direction probabilities and add the most probable one to the pa (lines 4-7). The set N holds most recently discovered neighboring cells of the pa. An important point to note is that the pa is added one neighboring cell at a time (line 8). This is necessary for discovering more probable cells among new neighbors (if any). The procedure continues until an upper limit on the number of cells S_{MAX} is achieved (line 3). The obtained paging area is a list of cells arranged in the decreasing order of $P_x(i)$. Note that if S_{MAX} is large, and if the same cell order is preserved, it is possible to obtain a paging area of an arbitrary size S relative to a same cell (by picking the first S cells of the list).

We define the largest paging area relative to a given cell as LPA (largest paging area). Then, given that the PACA has configured the LPA relative to a each cell, mobile hosts can obtain paging areas of individually defined sizes. Note that, for any chosen size S, the obtained paging area will be the subset of its LPA and will also have an optimum shape. Figure 9.4 illustrates three such paging areas pa_1, pa_2 and pa_3 relative to a same cell and of different sizes, where $pa_1 \subset pa_2 \subset pa_3 \subset LPA$.

9.3.3 Reducing the storage requirements

In order to avoid excessive memory consumption in paging agents, we make use of the paging area coding scheme described in Section 9.2.3 by defining layered paging areas as previously illustrated in Figure 9.4. Each layer comprises a number of cells encoded with the paging area coding scheme. For example, given an LPA of size $S_{MAX} = 50$; we define five paging areas pa_1, pa_2, ..., pa_5 ($pa_5 = LPA$) relative to a same cell and having the sizes 10, 20, 30, 40 and 50 (respectively). Then, each layer can be represented by 128 bits instead of 128×10 bits. As a result, a memory gain up to 10 is possible (if the LPA falls in a single cluster. This scheme however, reduces the granularity of paging area sizes.

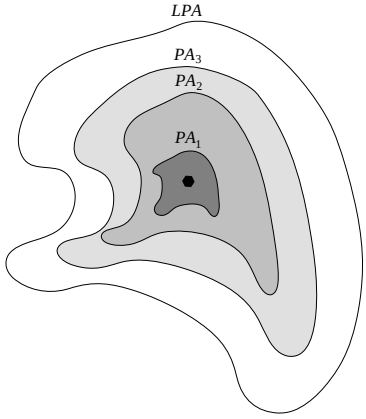


Figure 9.4: Paging areas of different sizes and relative to a same cell.

Procedure: **compose**(x)
00 $pa \leftarrow x$
01 $P_x(x) \leftarrow 1$
02 $N \leftarrow \{1\}$
03 **while** $|pa| < S_{MAX}$ *and* $N \neq \emptyset$
04 $N \leftarrow \emptyset$
05 **for each** $i, j \mid i \in pa, j \notin pa, P(i \rightarrow j) > 0$
06 $P_x(j) \leftarrow P_x(j) + P_x(i) \times P(i \rightarrow j)$
07 $N \leftarrow$ *all* $k \mid k \notin pa$ *and* $P_x(k) > 0$
08 $pa \leftarrow pa \cup k \mid k \in N$ *and has the largest* P_x
09 return(pa)

Figure 9.5: A procedure for composing a paging area relative to the cell x.

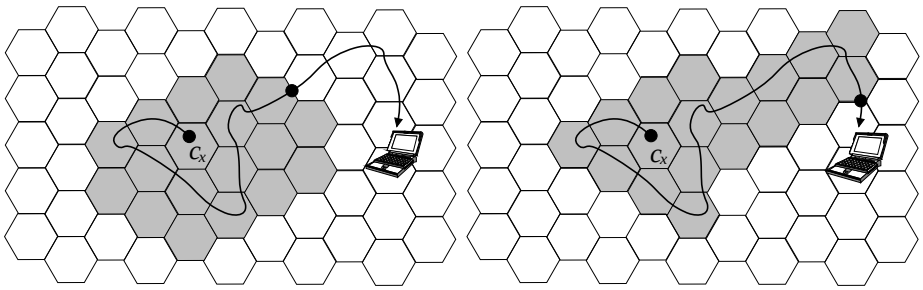


Figure 9.6: Two different paging area shapes (solid dots represent binding updates).

9.4 Paging area convergence analysis

9.4.1 Utilization rate of a paging area

In this section, we develop a measure for paging area shape efficiency.

We start with the following example illustrated in Figure 9.6: A particular mobile host arrives at cell c_x and requests its current paging agent a paging area pa_x relative to cell c_x. The paging area pa_x has a particular shape. For the moment, we assume that the paging area size S is constant ($S = 18$ in Figure 9.6). We denote the set of different cells that the mobile host visits before leaving pa_x as $\rightsquigarrow_x$, and call it a *track*. We denote the i^{th} track of the mobile host in pa_x as $\rightsquigarrow_x^i$.

Then, for a given track $\rightsquigarrow_x^i$ of the mobile host, we define

$$u_x^i = \frac{|\rightsquigarrow_x^i|}{S} \tag{9.1}$$

and say that the shape of the paging area pa_x is not efficient when u_x is small. Figures 9.6a and 9.6b illustrate two different paging area shapes which give $u_x^1 = 9/18$ and $u_x^2 = 13/18$, respectively. Clearly the paging area shape shown in Figure 9.6b, is more efficient since upon incoming session, the mobile host will be paged in a larger number of cells that it has visited. Furthermore, in this figure more binding updates are avoided by adapting the paging area shape to the mobile host's track.

If we generalize the above formulation, we can define the overall shape efficiency of pa_x as

$$\rho_x = \lim_{L\to\infty} \frac{1}{L} \sum_{i=1}^{L} u_x^i \tag{9.2}$$

where ρ_x is what we call the *utilization rate* of the paging area pa_x. L is the number of times that the mobile host visits and leaves pa_x. The utilization rate is bounded by

$$\frac{1}{S} \leq \rho_x \leq 1 \tag{9.3}$$

The lower bound is not zero because every track $\rightsquigarrow_x^i$ of the mobile host contains at least one cell which is c_x. On the other hand the upper bound can be achieved if and only if we have the probability $P\ (|\rightsquigarrow_x| = S) = 1$. Then, given that cells are numbered as $c_0, c_1, c_2, ...$, and if we define $P_x(c_i) = P(c_i \in \rightsquigarrow_x)$ it is easily shown that the utilization rate of pa_x is

$$\rho_x = \frac{\sum_{c_i \in pa_x} P_x(c_i)}{S} \tag{9.4}$$

where $P_x(c_i)$ is the probability that the mobile host will visit the cell c_i before leaving pa_x.

Note that the numerator of the utilization rate is the average number of different cells visited before leaving a paging area. Therefore we can reasonably define:

$$\rho = \frac{S_{eff}}{S} \tag{9.5}$$

where S_{eff} is the effective paging area size. S_{eff} can be used for comparing the efficiency of two paging area shapes proposed in a same set of conditions: same paging area size, same cell structures and same mobility pattern. However, as a measure, the utilization rate is more complete since it gives us also information about how much of a location area is "useful compared to its size".

If we define BG_x as the average number of movements of the mobile host in pa_x i.e., the average binding update gain of M in pa_x compared to Mobile IP, then we have

$$BG \geq S_{eff} - 1 \tag{9.6}$$

In other words, S_{eff} is our lower bound on binding update gain. As a result, the utilization rate is a reasonable measure of location area shape efficiency because:

If the utilization rate of a paging area is large, then

1. When there is an incoming call for the mobile host, the paging agent will page that mobile host in the cells that it has most likely visited.

2. The average rate of BUs that the mobile host sends upon leaving that paging area will be small (BG will be high).

In the context of the adaptive individual paging scheme, our objective when we choose paging area shapes is to reduce binding update rates as much as possible. Therefore we are more interested in the second item.

We will use the utilization rate concept for the convergence analysis of paging areas. By "convergence" we mean evolution of the paging area utilization rate in time.

9.4.2 Methodology

We define two concentric zones as illustrated in Figure 9.7. The sampling zone (SZ) is a small portion of the cellular network where we can assume a fixed mobility pattern. We evaluate the performance of the LPAs relative to the cells which are in the test zone (TZ). Note that an LPA relative to a cell in the TZ is not necessarily a subset of the TZ. Hence the need for two concentric zones. The size of the SZ is chosen large enough so that it can cover all subject LPAs. We set $S_{MAX} = 50$ for LPA sizes. We compute the utilization rates of the LPAs relative to each cell in the TZ (initially 1/50) using the Formula 9.2 with $L = 1000$ and $S = S_{MAX} = 50$. The overall performance is the average of the utilization rates. For simulation simplicity we use square shaped and equal sized cells.

We analyze four different mobility patterns:

- Random Walk Areas (RW): In each cell of the mobility zone, there are 4 possible directions that a mobile host may take. These are North(N), South(S), West(W) and East(E). The direction probabilities are equal and uniformly distributed over the sampling zone.

- Crossroads (4/2D): Cells are placed regarding the Manhattan grid model. We are interested by the LPA convergence at crossroads where a mobile host may take one of 4 directions with the probabilities described above. Mobile hosts change their directions only at crossroads. Therefore there are only 2 possible directions (N-S or W-E) in the cells connecting them, (hence the name 4/2D). Each crossroads is 10 cells away from the next one.

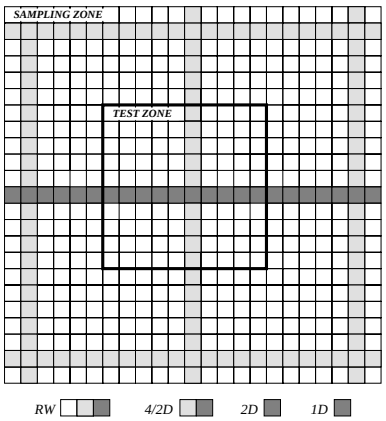


Figure 9.7: Mobility patterns and corresponding cells.

- Two-way Highways (2D): Cells are placed in a linear fashion. In each cell there are 2 and equally probable directions (W-E). A mobile host does not change its direction in the sampling zone.
- One-way Highways (1D): The cell structure is the same as 2D. However, there are only one possible direction (W) in each cell.

Figure 9.7 illustrates the cells that may be visited with each pattern. Both the SZ and TZ are square shaped. The size of the TZ is 10×10 cells. The size of the SZ is large enough to cover all LPAs. Each pattern is analyzed as a separate case study. In practice, user mobility is the combination of these patterns (and many others that we are not able to model). However, our goal is to analyze the convergence of LPAs in separate portions of the mobility area where we can reasonably assume a single, fixed pattern. An important point to note here is that these patterns are presented in increasing order of paging area utilization rates that we could obtain. Our goal is to show that the maximum utilization rate that can be obtain does not depend on algorithmic performance.

We generate a random pair of neighboring cells $[a, b]$ in the SZ (in accordance with the mobility pattern) and count the frequency of the corresponding direction in cell a in order to compute the probability of moving to cell the b. This procedure simulates a single sample sent along with a binding update message of a mobile host. We continue until the LPAs converge.

9.4.3 Results and Discussion

Figure 9.8 shows the convergence of LPAs as a function of number of samples. Each received sample is processed in order to update the direction information in the indicated cell and the LPAs in the TZ. As the direction information becomes more accurate, the utilization rates of the LPAs increase. After the reception of a certain number of samples (which depends on the mobility pattern), the LPAs shapes remain constant. At that point further sampling is useless. All LPAs converge together since the same direction probabilities are used for the convergence of more than LPAs. However, in practice user flow rate may not be uniform and some of the LPAs may converge later than others.

The maximum utilization rate that can be obtained (upon convergence) does not depend on algorithmic performance. Full utilization can be only obtained with the 1D pattern because this pattern is the only one which gives $P\ (|\rightsquigarrow_x| = S) = 1$.

The RW pattern is interesting because this pattern is the most difficult pattern to deal with since in this case one needs more samples (in each cell of the SZ, there are four equally probable directions). The SZ size for this pattern is $50 \times 50 = 2500$ (large enough to cover all LPAs in the

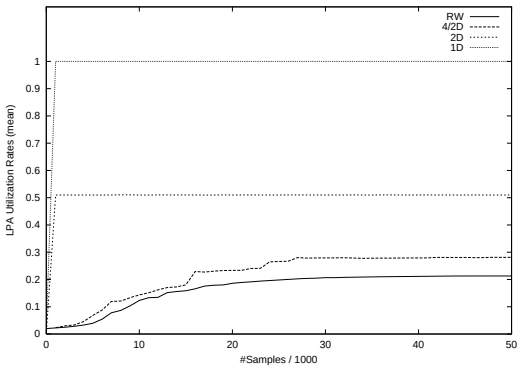


Figure 9.8: Mean of the LPA utilization rates as a function of number of samples.

TZ). Simulation results show that in this "worst" case, a paging agent will need 40,000/2,500=16 samples per cell in order to have all LPAs in its domain converged.

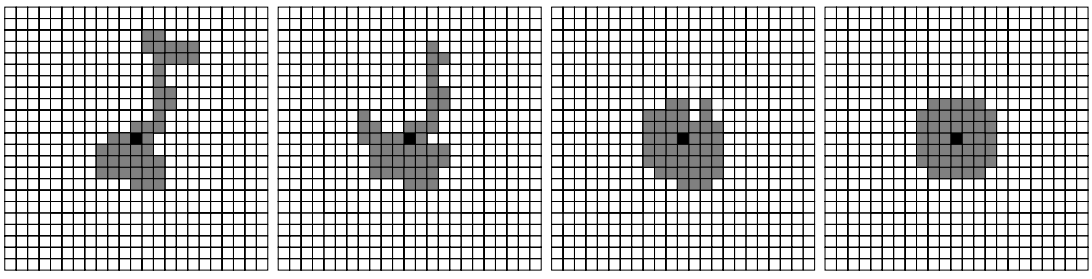
We assume the following parameters:

$$\#\text{esers/cell} = 50$$
$$\#\text{binding update/user/hour} = 1$$
$$\text{sampling rate p= \%0.5}$$

Then, the LPAs in the paging agent domain can converge in:

$$\frac{16}{50 \times 1 \times 0.005} \; = \; 64 \; hours \tag{9.7}$$

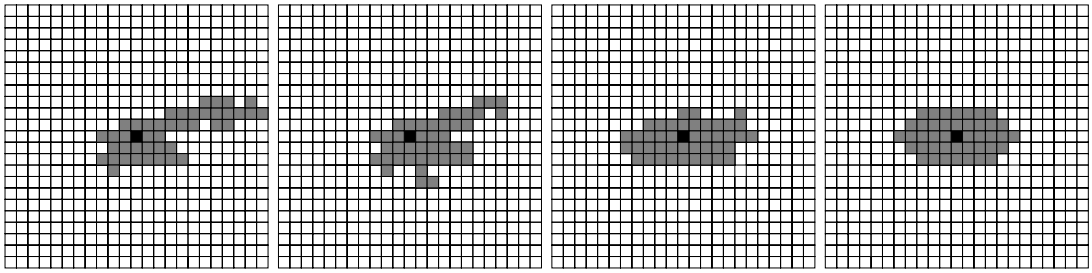
This is a promising estimation although we have made weak assumptions. In practice both the number of users and binding update rates are higher. The number of samples that is needed for convergence, depends on host mobility characteristics. For example, along a one-way highway where all mobiles move in the same direction (i.e. the direction entropy is zero), 1 sample/cell will be enough. When host mobility pattern is less predictable (in urban areas for example), more samples will be needed.



(a) $\rho = 0.107, BG = 4.6$ (b) $\rho = 0.115, BG = 7.2$ (c) $\rho = 0.220, BG = 13.7$ (d) $\rho = 0.235, BG = 16.5$

Figure 9.9: RW pattern.

Finally, we analyze the effect of the paging area shape optimality on BG (binding update gain compared to Mobile IP). Figures 9.9 and 9.10 show example LPA shapes, their utilization rates



(a) $\rho = 0.135, BG = 4.9$ (b) $\rho = 0.154, BG = 6.1$ (c) $\rho = 0.228, BG = 10.2$ (d) $\rho = 0.240, BG = 15.6$

Figure 9.10: Modified RW pattern.

for two different mobility patterns, and the resulting binding update gains that can be obtained compared to Mobile IP. In Figure 9.9, we analyze the RW pattern. On the other hand, in Figure 9.10, we analyze a modified RW pattern (the probabilities of the directions N, S, E, W are 0.1, 0.1, 0.4 and 0.4 respectively). We use two paging area sizes 45 and 43 in order to have a symmetric paging area shape view relatively to a same cell (the black one). This helps our visual understanding of the results. In these figures, there are two important points to note. First, as the shape of a paging area becomes optimal its utilization rate increases. Second, important gains BU gains can be obtained by finding optimum paging area shapes.

Chapter 10

Discussion and conclusion

Contents

10.1 Hash-based IP paging

Hash-based IP paging allows the paging sub-system (paging agent or access router in the two models that we have analyzed) to page multiple idle hosts using a single message. Several host addresses waiting the paging queue are represented using a Bloom filter. This leads to a small false location update probability. The total number of paging and false location updates is smaller than standard paging, hence bandwidth is saved. The cost to pay is hash computation cost (link layer interrupts will be discussed below) by the paging system or access points.

We have analyzed two IP paging modes: with and without time-slotted dormant mode support. Time slotted dormant mode is an efficient power saving mode that used in current cellular systems. When time-slotted dormant mode is not available, Bloom filters can be applied by the paging sub-system. Depending on the paging area size (in number of users) bandwidth can be saved or lost in the core network. However, wireless bandwidth is saved in each cell, regardless of paging area size. With time-slotted dormant mode support, Bloom filters can be applied by access points which need the identifier of a paged host to determine its time-slot. In this case access points can also filter the false location updates and the bandwidth gain is 1 in the core network.

In our simulations, we have rate limited the paging messages, in order to exploit the bandwidth gain of hash-based IP paging. In each simulation we have increased by the paging area size by 1,000 users. A larger paging area covers a wider geographical area, which further reduces the binding update rate and battery drain on mobile hosts. With rate limiting, standard paging could not resist against large paging areas, excessive paging delays occurred. Using hash-based paging we have obtained much smaller queueing delays at hash computation and link layer interrupt cost.

10.1.1 Do we care about link layer interrupts?

In IP paging, a paging message is an IP packet. As a consequence, a paging message needs to be passed to the IP layer by the idle hosts in the paging area. Unless a specific solution is available, an idle host's IP module will be interrupted upon every incoming session in its paging area. Previous work never mentioned this problem. Currently it is not clear if the processing of

a paging message by the IP module is significantly more expensive than processing it by the link layer interface.

If 1 idle host is paged at a time, link layer interrupts can be avoided using link layer multicast. A technique that is similar to solicited node multicast address in IPv6 can be used[3]. When multiple idle hosts are paged using a single paging message, the paging packet must be broadcasted. As a consequence, all hosts in the paging area will be interrupted (only idle hosts, if special paging channels are available). Hash-based IP paging suffers from this problem. However, any paging mechanism (hash-based or not) that incorporates the concurrent paging optimization suffers from the same problem. By inserting multiple identifiers to the paging message (i.e. increasing the message size instead of using a Bloom filter), current cellular systems are able to page multiple terminals. P-MIP (that we have previously reviewed) deploys the same technique, by inserting multiple home addresses in a IP paging messages, hence suffers from the same problem.

10.1.2 Comparison with address resolution

The similarity of IP paging and address resolution is very visible (we have applied the same optimization to both of them).

One difference is that in address resolution access points are capable of wireless bridging. Another difference is performance. In address resolution, caching is possible. An idle mobile host can change its cell, but if its neighbor cache is still valid, the access router does not need to perform address resolution. One the other hand, address resolution is less efficient than IP paging, in terms of energy. Upon link layer handover (within subnet boundaries) an idle mobile host needs to re-associate with its current access point, which is not the case with IP paging.

10.2 Adaptive IP paging architecture

We have developed an adaptive IP paging architecture, in which paging areas are represented as "data". This approach allowed us some paging area design flexibility, i.e. to dynamically change the paging area sizes and shapes. The advantage of this scheme is a more optimal paging system than using static paging areas since each host has a different mobility and incoming session rate.

Other algorithms and architectures than the ones developed in this thesis may be possible. For example, one can think about a paging area configuration algorithm that is run by access points in fully distributed manner. However, in this case the sampling process (for obtaining the movement direction probabilities) may be very difficult. For this reason, we have defined a very centralized mechanism. In the proposed architecture, there is one Paging Area Configuration Agent (PACA) per a large cellular domain, which is responsible for sampling and paging area configuration in its domain. Clearly, this approach has the disadvantage of any centralized scheme, i.e. considerable processing and storage costs and low reliability. On the other hand, such a centralized approach is the one that allowed us to avoid the complexity and signaling redundancy in the remaining and basic network elements such as access points.

Note also that we have assumed a very restricted subnet/cell mapping in which subnet and cell boundaries are identical. However, the same architecture and sampling algorithm can be easily adapted to large subnets with may cells.

Part III

End-to-End IP paging

Chapter 11

End-to-end IP Paging

Contents

11.1 Introduction

In the previous chapters we have reviewed and optimized previous IP paging proposals, where IP paging was a network-based service for reducing signaling costs in the network and power drain on mobile hosts. In this chapter, we develop and analyze a completely different IP paging approach to address a very different problem. We propose an End-to-End IP Paging (E2EP) protocol to cope with home agent failure or unavailability in MIPv6.

E2EP replaces the paging area concept by a large *Living Area*. A living area is a set of IP subnets that covers a wide geographical area. Our basic motivation is that most mobile users have a more or less fixed living area that they rarely leave. Living areas can be manually or dynamically configured. A mobile host can run a host-based algorithm to learn its living area and distribute it to its critical correspondent nodes (e.g. family, friends and colleagues, or the other members of a local organization etc.). A critical correspondent nodes can contact the mobile host by paging it in its living area, if the mobile host's home agent is temporarily down or unavailable. The paging procedure consists of sending copies of a session initiating packet to the subnets covered by the mobile host's living area. By configuring and distributing a living area, a mobile host avoids sending binding updates to every critical correspondent node, upon movement within its living area.

Each E2EP-enabled mobile host has its own (generally only one) living area which covers the regularly visited places (e.g. home, shopping centers, cafés, etc.). Thus, a living area is not only very large, but it also has an optimal shape that is adapted to host mobility. The size and shape of a host's living area, reduces as much as possible the probability that the host will leave its living area. Mobile hosts that do not have a fixed living area (i.e. that make frequent and unpredictable long-distance movements) can not profit from E2EP. The E2EP protocol is rather useful for mobile hosts that rarely leave their living area, or for mobile hosts that serve a particular goal and not supposed to leave their living area (such as ambulances).

In summary E2EP reduces the home agent dependency with the following constraints: First, it is useful for mobile hosts that rarely leave their living area (which covers an important portion

of mobile hosts). Second, its use is limited to previously known critical correspondent nodes that are explicitly sent the living area information.

11.2 Motivations

11.2.1 Home agent failure

E2EP is useful for critical correspondent nodes when a destination mobile host is unreachable because its home agent (the only entity to know its location) is down, unreachable or fails to route packets.

MIPv6 improves home agent reliability by recommending the deployment of multiple home agents. When a home agent crashes, another home agent continues to provide service. However, currently there is no evidence that such a solution can recover from all failures. Such a fault recovery may also take time depending on the source of failure. In the mean time a critical incoming session may fail. Furthermore, we are not even certain that the home agent service will always be well-administrated, hence such an expensive solution may not be always applicable. As noted below, the home agent service has implications on privacy. Hence, some users may prefer running personal home agents on their loosely administrated personal computer with permanent Internet connection.

Using E2EP, a correspondent node -itself- can locate a destination mobile host when the home agent fails. Provided that a mobile host rarely leaves its living area, there is a significant chance that E2EP can succeed. Therefore, E2EP is worth trying if a correspondent has no other way to contact a mobile host and especially if the session is critical.

11.2.2 Location privacy

In MIPv6, home agent is designed to offer guaranteed reachability regardless of user location (which is not guaranteed as explained above). This service does not come for free. It requires that a mobile host reports its location to the system upon "each movement". For example, when a host enters a hot-spot it must report its location by sending a binding update. Otherwise, the host may not be able to receive incoming sessions (if connectivity with the outdoor cellular link is not available or lost, for example).

One possible solution is to install a personal home agent at home on a personal computer with a permanent Internet connection. However, this may not be always possible.

Using E2EP, a mobile host can use home agent service only when necessary: i.e. upon leaving its living area. In order to hide its location from its home agent, a mobile host can stop sending binding updates to its home agent, and be reachable to its critical correspondent nodes by E2EP (provided that the user is not planning to leave her living area).

11.2.3 Limited MIPv6 without home agent

Using E2EP, a number of mobile hosts may form a MIPv6 community and communicate without home agent in their living area. Each host can configure a living area, distribute it to other members of the community and be reachable using E2EP. Alternatively, some mobile hosts may have the same living area which can be manually configured on the subject mobile hosts.

Complete (or, near to complete) home agent independence may be possible if these mobile host serve a specific goal and not supposed to leave their living area. The participants of a city fire department, or ambulances are possible examples. These nodes may use MIPv6 and profit from the access infrastructure for free but do not have (do not need, do not trust, or do not rely on) home agent service. Note that these are critical mobile hosts, and home agent failure may have serious consequences.

Chapter 12

Protocol description

Contents

12.1 Introduction

In this chapter, we develop an E2EP protocol that does not require any modification to the routing infrastructure. A minor modification to the access router code is desirable, in order to support correspondent node well-behavior. This issue will later discussed in Section 14.3.

E2EP should be viewed as a variant of MIPv6. E2EP is only used to locate a destination mobile host. Mobile hosts have a fixed home address and configure a care-of-address upon movement. During a session, mobile hosts use MIPv6. They can change their care-of-address, however upper layer connections will not be affected because standard MIPv6 is used.

12.2 Terminology

- MH: Mobile host.
- LA: Living Area. A set of subnets that covers a user's living area.
- LA_{MH}: Living Area of a mobile host MH.
- Q: Size of a living area (in number of subnets).
- pre_i: Prefix of subnet i.
- ID_{MH}: Interface identifier of a mobile host.
- $|$: concatenation.
- CCN: Critical Correspondent Node.

- $CCNlist_{MH}$: CNNs of a mobile host MH.
- P_1: Session initiating packet.

12.3 E2EP modes of operation

E2EP can be used to locate a mobile host (provided that its has not left its living area) in case of home agent failure or if the mobile host does not have any home agent. We define two different E2EP modes to handle these cases separately.

In **HAF (Home Agent Failure)** mode, the critical correspondent node CNN tries to contact the mobile host MH by sending a packet to its home address (i.e. using regular MIPv6). Normally the mobile host's home agent should intercept and forward the packet to the mobile host's current location. If MH does not respond or home agent failure is detected, CNN tries with E2EP. The HAF mode illustrated in Figure 12.1. We do not specify any technique for reliably detecting home agent failure. Different mechanisms may be possible.

In **UHA (Unavailable Home Agent)** mode, the mobile host MH does not have any home agent. In order to establish a session with MH, CNN directly proceeds with E2EP.

12.4 Living area registration

The mobile host MH has a living area $LA_{MH} = \{pre_1, pre_2, ..., pre_Q\}$ that covers Q subnets, and a list $CCNlist_{MH}$ of its critical correspondent nodes. The members of $CCNlist_{MH}$ may or may not change in time, depending on the mobile host's policies and objectives. The living area LA_{MH} can be either manually or dynamically registered to the members of $CCNlist_{MH}$. Dynamic living area registration is achieved by sending a binding update to CNN, with the following simultaneous care-of-addresses:

$$pre_1|ID_{MH}$$

$$pre_2|ID_{MH}$$

.

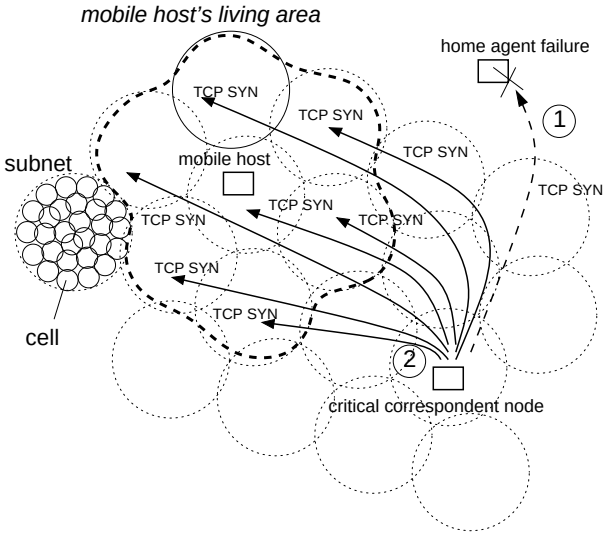
.

.

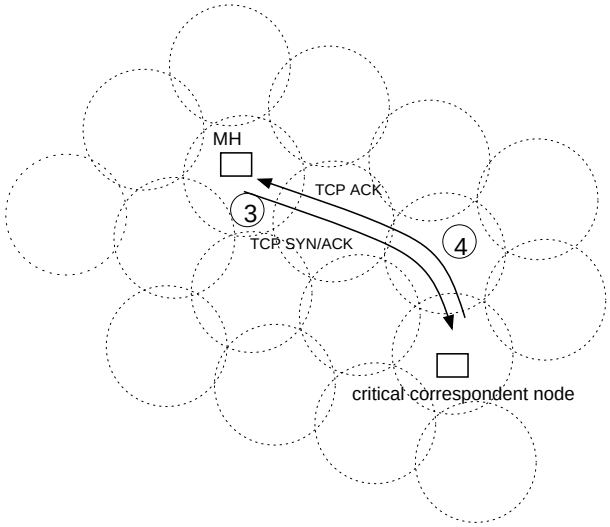
$$pre_Q|ID_{MH}$$

where $pre_1, pre_2, ..., pre_Q$ are the subnets covered by the LA_{MH}. The binding update also sends the E2EP mode (HAF or UHA) that should be used by the correspondent node. In the case of the manual living area registration, LA_{MH}, ID_{MH} and the E2EP mode are manually configured by the members of $CCNlist_{MH}$.

The living area needs to be stored in non-volatile storage by the members of $CCNlist$. When living areas are dynamically configured, they can be dynamically sent to a new CNN, i.e. the $CCNlist$ of a mobile host may grow as time passes. Manual living area configuration is rather suitable for a group of mobile hosts that serve a particular same goal, their living area does not change, and they are not supposed to leave their living area. These mobile hosts may use the HAF or UHA mode, which is independent of how living areas are configured and distributed. Figure 12.2 shows E2EP operation in a manually configured living area in UHA mode. All mobile hosts that from the special purpose group are manually configured with the same living area.



(a)



(b)

Figure 12.1: HAF (Home Agent Failure) mode. E2EP for establishing a critical TCP connection in case of home agent failure (ICMP destination unreachable messages are not shown). E2EP is useful, because it is "likely" to succeed, since the the mobile host rarely leaves its living area. Note also that some mobile hosts are not supposed to leave their living area.

12.5 Idle movements inside a living area

While MH moves within its living area, it monitors the router advertisements to detect its movements and configures a care-of-address $pre_i|ID_{MH}$ each time it performs MIPv6 handover in its living area (where $pre_i \in LA_{MH}$). When idle (i.e. not in active communication), the MH does not send binding updates upon moving in its living area. This allows the MH to conserve energy and avoid consuming bandwidth for sending binding updates to all correspondent nodes (in the $CCNlist_{MH}$ list) upon each movement.

12.6 Incoming session establishment

In HAF case, a critical correspondent node CCN can establish a session with MH using E2EP, provided that MH has not left its living area. The CCN first tries to contact MH using standard MIPv6. If home agent failure is detected, CCN performs E2EP. In UHA mode, CCN directly performs E2EP.

The E2EP procedure consists of sending Q copies of a packet P_1 (session initiating packet, e.g. TCP SYN) to the addresses:

$$pre_1|ID_{MH}$$

$$pre_2|ID_{MH}$$

$$.$$

$$.$$

$$.$$

$$pre_Q|ID_{MH}$$

which is called end-to-end IP paging.

The packet P_1 is inserted the mobile host's home address in a Type 2 routing header defined by MIPv6. The mobile host receives the packet P_1 in one of the subnets, sends a binding update to the correspondent node, and the communication begins.

It is important to note that a mobile host may not have a home agent, but it has a globally unique home address.

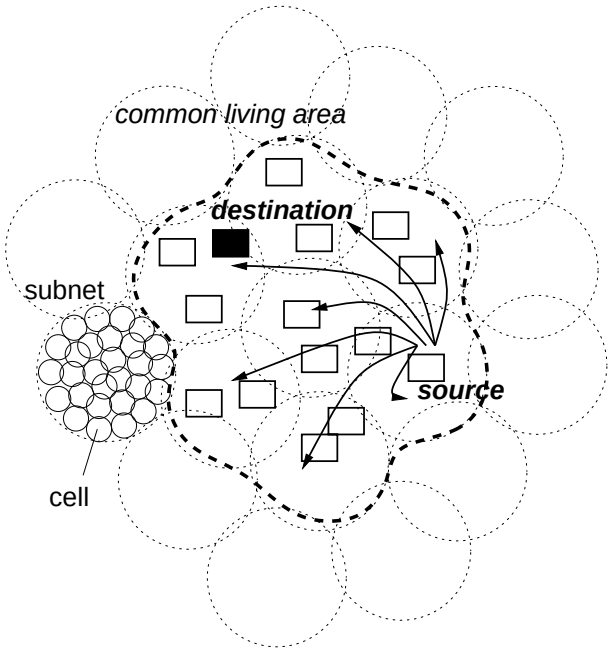
12.7 Movements during a session

During a session, MH and CCN behave as defined by standard MIPv6. Recall that, the MH has a fixed home address. Outbound packets and inbound packets carry its home address. Upon moving, the mobile host can configure a care-of-address with any ID, but a binding update has to be sent to the correspondent node.

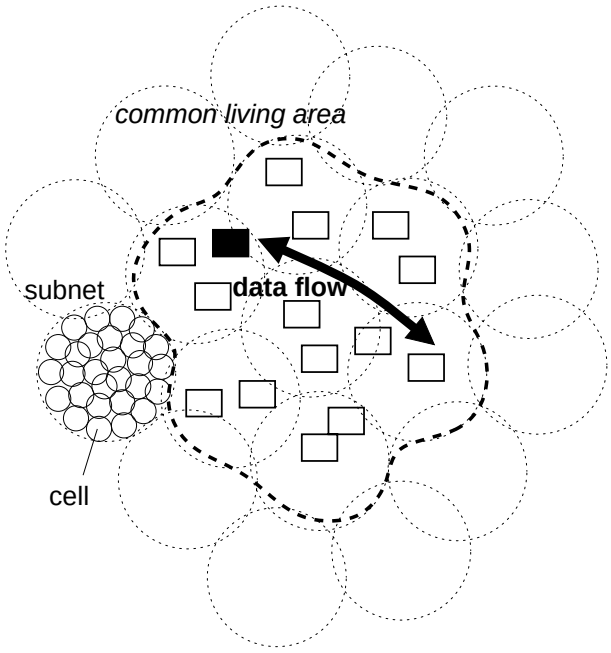
As a result, upper layers are not affected by changing care-of-addresses, since regular MIPv6 is used.

12.8 Duplicate addresses

In the above protocol, a mobile host's interface identifier is fixed and distributed to critical correspondent node's in advance. Upon movement, the mobile host MH will perform DAD, by multicasting a neighbor solicitation for its tentative interface identifier. If a neighbor advertisement is received, the mobile host must retry with another identifier. In this case, however, the mobile host will become unreachable to its critical correspondent nodes, since interface IDs are distributed in advance.



(a)



(b)

Figure 12.2: E2EP in a static living area (UHA mode). All mobile hosts are configured with the same living area, which they are not supposed to leave.

In order to address this problem, the mobile hosts can configure and distribute more than one interface IDs along with their living area. In this case, the mobile host MH will still have 1 living area $LA = pre_1, pre_2, ..., pre_Q$, but for example 3 different random IDs $ID^1_{MH}, ID^2_{MH}, ..., ID^3_{MH}$ which are distributed to all members of $CCNlist$.

By default, the mobile host always uses ID^1_{MH}. If ID^1_{MH} collides, the next ID is tried i.e. ID^2_{MH}. In case of two consecutive collisons the mobile host can configure ID^3_{MH} etc. If IDs are randomly generated as proposed in [26] address collision probability will be very low as discussed in Section 6.6.

Let's assume that MH suffered from 1 collision in its current subnet and ID^2_{MH} was successfully configured. Then, a critical correspondent node CCN wishes to communicate with MH, but its home agent is temporarily down. The CCN tries first by transmitting a packet to the following care-of-addresses:

$$pre_1|ID^1_{MH}$$
$$pre_2|ID^1_{MH}$$
$$.$$
$$.$$
$$.$$
$$pre_Q|ID^1_{MH}$$

which are not received by MH which listens on ID^2_{MH}. Upon timeout (and/or a few trials with ID^1_{MH}), CCN retries with the following care-of-addresses:

$$pre_1|ID^2_{MH}$$
$$pre_2|ID^2_{MH}$$
$$.$$
$$.$$
$$.$$
$$pre_Q|ID^2_{MH}$$

The mobile host MH receives the packet, and the communication between the hosts begins using standard MIPv6. During the session binding updates are sent to the correspondent node, hence any ID can be configured upon movement as defined by standard MIPv6. At the end of the session, however, the mobile host will enter idle mode and must configure the default ID ID^i_{MH}, where $i = 1$. In case of collision, i must be incremented.

Chapter 13

Dynamic living area configuration

Contents

13.1 Introduction

In this chapter we develop a simple algorithm for dynamically configuring a living area. The algorithm is basically the same as the one that we have proposed for adaptive IP paging (Chapter 9). The algorithm that we define here differs in the sampling policy. In E2EP, the mobile host learns its own personal living area by sampling its own movements, hence living area shape is personal (i.e. not aggregated like in Chapter 9).

A mobile host can dynamically learn the set of subnets that are regularly visited in its living area, as well as the movement direction probabilities in each subnet. Using this information, a contiguous living area can be configured and sent to critical correspondent nodes.

13.2 Algorithm

Consider, for example, the following IPv6 mobility history of a a mobile host:

$$GTYUIUYTGBGFDERTGFVBGHNJUYTG \rightarrow$$
$$\rightarrow TYHGFGHJUYTRFGBGFRTGTREDFGHY$$

where each letter represents a subnet prefix.

The simplest information found in this history is adjacent subnets (geographically or vertically). For example, the subnets $[G,T]$ $[T,Y]$, $[Y,U]$ are adjacent subnets.

Another valuable information found in the history is the MIPv6 movement direction probabilities. In the above history, the following movements are made in the subnet G: 3 $[G \rightarrow T]$ movements, one $[G \rightarrow H]$ movement, 2 $[G \rightarrow B]$ movements and 4 $[G \rightarrow F]$ movements. I.e, a total of 10 samples, which gives the following movement direction probabilities in subnet G:

$$p([G \rightarrow T]) = 3/10$$
$$p([G \rightarrow H]) = 1/10$$
$$p([G \rightarrow B]) = 2/10$$

$$p([G \rightarrow F]) = 4/10$$

The above mobility history is not long enough, and the movement direction probability information is only approximative. As time passes, however, more and more samples can be collected and more accurate probabilities can be obtained.

A mobile host can run a simple algorithm to configure an optimal shaped living area regarding the movement direction probabilities in its living area. A small living area that the mobile host can configure is $[G, T, H, B, F]$. This is a living area of size $Q = 5$ and its center is the subnet G. Below we show how a larger living area can be configured. The movement direction probabilities in subnets T, H, B, F are:

$$p([T \rightarrow Y]) = 2/6$$
$$p([T \rightarrow R]) = 2/6$$
$$p([T \rightarrow G]) = 4/6$$

$$p([H \rightarrow N]) = 1/4$$
$$p([H \rightarrow G]) = 1/4$$
$$p([H \rightarrow J]) = 1/4$$
$$p([H \rightarrow Y]) = 1/4$$

$$p([B \rightarrow G]) = 1$$

$$p([F \rightarrow D]) = 1/6$$
$$p([F \rightarrow V]) = 1/6$$
$$p([F \rightarrow G]) = 3/6$$
$$p([F \rightarrow R]) = 1/6$$

respectively, as illustrated in Figure 19.8. The next subnet that we add to this living area must minimize the probability that the host will leave the living area. The candidate subnets that may be added are the subnets R, D, V, J, N, Y (i.e. the neighbors of the living area $[G, T, H, B, F]$). Upon leaving the living area $[G, T, H, B, F]$ the subnet R will be visited with the largest probability which is:

$$p([G \rightarrow F]) \times p([F \rightarrow R]) + p([G \rightarrow T]) \times p([T \rightarrow R]) = 1/10$$

thus the subnet R is added to the living area to obtain the living area $[G, T, H, B, F, R]$ of size $Q = 6$. This is denoted "first living area" in Figure 19.8. It can be augmented using the same procedure recursively to obtain the living area "augmented living area".

The above algorithm is formalized in Figure 13.2, which is an adaptation of the algorithm in Figure 9.5 to the living area concept.

Note that when subnets are large, a small sized living area can guarantee that a mobile host will rarely leave its living area. We do not argue that every mobile host will perfectly succeed. The living area of a mobile host may not perfectly overlap with the geographical area where the user lives. However, some reasonable success can be expected in most cases. Upon home agent failure, the mobile host is likely to be located in its living area. It is the mobile host's responsibility to configure better living areas as time passes.

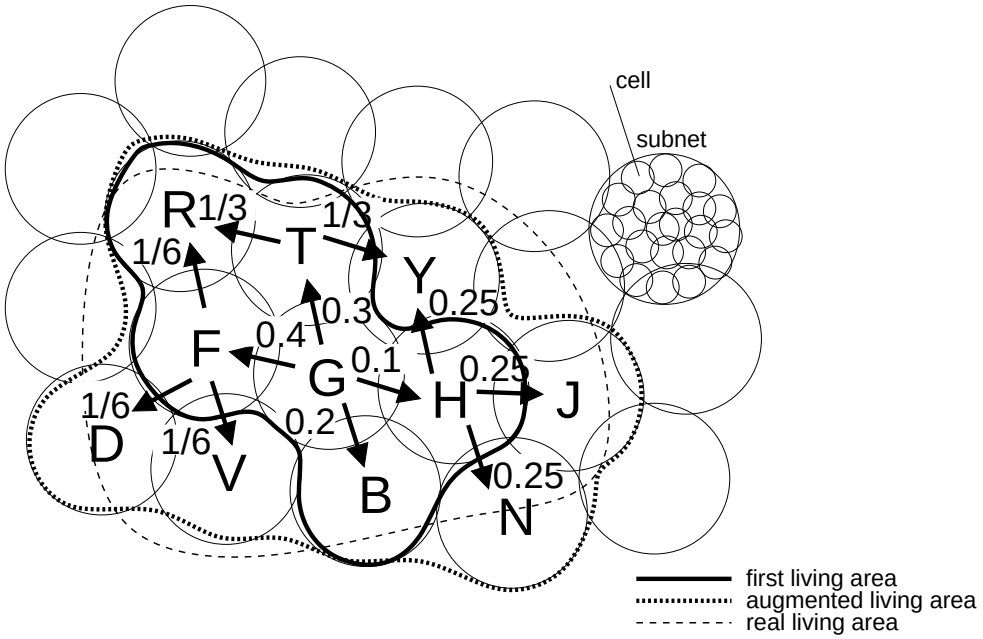


Figure 13.1: Living area configuration example.

Procedure: **configure**(x)
$LA \leftarrow x$
$P_x(x) \leftarrow 1$
$Q \leftarrow 1$
$N \leftarrow \{1\}$
while $Q < Q_{MAX}$ *and* $N \neq \emptyset$
    $N \leftarrow \emptyset$
    for each $i, j \mid i \in LA, j \notin LA, p(i \rightarrow j) > 0$
        $p_x(j) \leftarrow p_x(j) + p_x(i) \times p(i \rightarrow j)$
    $N \leftarrow$ *all* $k \mid k \notin LA$ *and* $p_x(k) > 0$
    $LA \leftarrow LA \cup k \mid k \in N$ *and has the largest* p_x
    $Q \leftarrow Q + 1$
return(LA)

Figure 13.2: A procedure for configuring a living area relative to a subnet x.

13.3 Memory and battery consumption

The living area configuration algorithm that we have defined does not require that a mobile host record all its IPv6 mobility history. Movement direction probabilities in each subnet of a living area can be updated periodically regarding the most recently collected samples (e.g. during a day or week).

The living area configuration algorithm shown in Figure 13.2 can be run periodically (e.g. once per day or week) and possibly when the mobile host is plugged. As a result, it does not necessarily represent a battery consumption overhead.

Chapter 14

Discussion and conclusion

Contents

14.1 Home-based location management vs end-to-end IP paging

In Mobile IPv6 (MIPv6), a mobile host blindly relies on home agent service for receiving incoming sessions. While away from home, the mobile host keeps informed its home agent when it moves, by sending a binding update message. Binding update messages update the binding between the mobile hosts' publicly known global identifier (home address) and the care-of-address. The care-of-address combines two information: location (subnet identifier) and a local identifier (interface identifier) that identifies the mobile host in its current subnet. The home agent is a special router located in the home subnet of the mobile host, i.e. the subnet where the mobile host's publicly known home address points. A correspondent node that wishes to communicate with the mobile host, sends its packet to the mobile host's home address. Since the home agent is always kept up to date about the current subnet of the mobile host, it can forward the correspondent node's packet to the mobile host's care-of-address.

The following definitions will help us better understand the above location management technique:

- Location space: The set of subnets where the mobile host is reachable to its potential correspondent nodes.
- Correspondent node space: The set of correspondent nodes that can reach the mobile host.

Both location and correspondent node spaces are infinite in MIPv6, at the cost of home agent dependency. The mobile host is reachable to a correspondent node regardless of its current subnet. The mobile host is also reachable to any correspondent node, since it is contacted using its fixed and publicly known home address. However, the home agent in the case, is a single point of failure. When the home agent fails, both subnet and correspondent node spaces "collapse" and the mobile host becomes unreachable. This is unfortunate, because in most of the cases there may be a [correspondent node→ mobile host] path that is different and possibly shorter than the path [correspondent node→ home agent → mobile host].

End-to-End IP Paging (E2EP) is an alternative to the home agent service in much more constrained but the most useful location and correspondent node spaces. By reducing the location space to the mobile host's *living area*, and reducing the correspondent node space to the set of *critical* (i.e. known) correspondent nodes we can obtain an alternative to the home-based location management, which is E2EP.

A mobile host can learn its living area and send it to its critical correspondent nodes. In case of home agent failure, a critical correspondent node can -itself- page the mobile host in its living area. Provided that the living area is large enough and the mobile host rarely leaves its living area, the critical session can be established. I.e., E2EP is useful (especially if the session is critical) because it is likely to succeed in most of the cases. Most users do have a more or less fixed living area, which there rarely leave. Some mobile hosts such as ambulances that serve a well-defined area, are not supposed to leave their living area.

The network's participation to E2EP is limited to routing packets to their destinations, hence extra complexity is avoided. The algorithmic complexity and state is rather pushed to the edges of the Internet. The mobile end needs to learn its leaving area, i.e. a contiguous set of outdoor subnets that covers its living area. A mobile host can learn its living area using a slightly intelligent algorithm that is input the cumulative IP mobility history, extracts the movement direction probabilities and constructs a living area.

14.2 Non-standard use of E2EP

Non-standard E2EP is possible. E2EP does not require any modification to the network. Although optimizations are desirable for its proper operation (please see below), it can also work without any modification. Living area state is maintained by correspondent nodes. Paging does not require support from the network, a mobile host is paged by sending several copies of a packet to different subnets in its living area.

Thus, we suspect that E2EP may be implemented even if it is not standardized. For example, one can imagine a MIPv6 implementation X that supports a non-standard version of E2EP. Clearly, there is no interoperability problem between two hosts that run the same implementation. Then, Alice and Bob may prefer the implementation X because they can communicate without home agent (for privacy reasons for example). Alternatively, all ambulances in a city may run the implementation X because they wish to avoid the consequences of a home agent failure.

14.3 Helping the correspondent node well-behavior

Upon home agent failure, a well-behaving correspondent node should not desperately flood the Internet for locating a mobile host. A well-behaving node should send its packet to a subnet in the living area and "wait" for the mobile host's response, or a negative acknowledgment from the access router. Until a negative acknowledgment or the mobile host's response is received, the correspondent node can retransmit (by backing off) its packet to the same care-of-address, just like a regular host would do. Only when a negative acknowledgment is received, the correspondent node should retry in another subnet.

This negative acknowledgment is the ICMP *Destination Unreachable* message which may be, unfortunately, returned too late by current access router implementations. This would result in excessive session setup delays, penalizing the above defined well-behavior.

When IP hosts move, the meaning of host unreachability somewhat changes. By current practice, the term "host unreachability" is used to indicate that the subject host is disconnected from the Internet. In a wireless Internet, however, the host may be reachable, but in another subnet. While this confusion does not have serious implications in traditional IP mobility management, in E2EP it does. In E2EP, if a mobile host does not respond in one subnet, we tend to believe

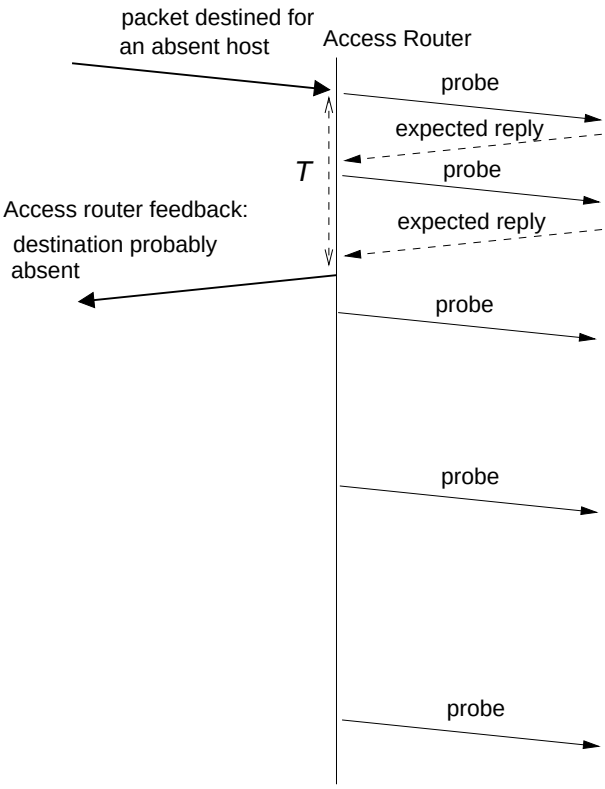


Figure 14.1: Fast access router feedback. In E2EP, if the mobile host does not quickly respond in one subnet, we tend to believe that it is in another subnet (i.e. absent) rather than saying it is unreachable. Thus, an early access router feedback can effectively increase the E2EP speed. Upon receipt of feedback, the mobile host can be paged in another subnet of its living area (in the mean time). In practice, the feedback message can be an ICMP Destination Unreachable message, since this is a soft error and does not induce connection closure on regular Internet hosts.

that it is another subnet rather than saying it is probably disconnected from the Internet. In the following discussion, when necessary, we will adopt the term "absent" instead of unreachable.

The ICMP destination unreachable error is a soft error. Upon receipt of such a message a host does not close the connection (unlike the port unreachability error). Then, we propose that an access router returns an early destination unreachable message, if the destination host does respond quickly. The access router will continue to try in order to make sure that the host is indeed absent. Since the mobile host is more likely to be in another subnet, such a quick indication will allow the correspondent node to effectively continue E2EP in the mean time. If the quick destination unreachable (absent) indication was premature, the mobile host will send a late reply to its access router, and receive the correspondent node's packet. In this case, the mobile host will reply to the correspondent node, while it is being paged in another subnet. If the mobile host's late reply to the correspondent node is lost, the paging procedure must be unnecessarily repeated by the correspondent node. While packet loss may occur, this is rather an exception. As a consequence, we prefer increasing the E2EP speed by assuming good network performance, rather than unnecessarily slowing down E2EP for dealing with packet loss.

This protocol is illustrated in Figure 19.10. We do not define any new message, we only shift (backwards) the time an ICMP destination unreachable message is returned. Since the host is absent, the destination IP address of the received packet is not in active use. Then the access router probes the destination address, rather than blindly routing the packet. The key to the quick

probe (and, hence quick feedback to the correspondent node) is the detection of host inactivity by the access router. This is rather an implementation detail.

The early indication increases the speed of E2EP, as mentioned above. The correspondent node will send its session initiating packet to the first subnet of the living area, receive an ICMP Destination Unreachable message, send the packet to another subnet of the living area, etc. For example, we assume a very large living area with $Q = 50$, a relatively long $T = 100$ms and also an average [access router<->correspondent node] round-trip time of 100ms (equivalent of the daytime round-trip time between Grenoble and a web server in San Francisco, 19 hops). With these pessimistic assumptions, E2EP can succeed in $\frac{(1+50)}{2} \times (100 + 100) = 5,100$ milliseconds, i.e. 5.1 seconds on the average. This may increase if some packets are lost in the Internet as mentioned above. However, we tend to believe that even 1 minute delay is better than nothing if the home agent, or home network is unreachable during a longer time period.

Part IV

Defending against denial-of-service attacks in Mobile IPv6

Chapter 15

Introduction

Contents

15.1 Review of Mobile IPv6 security

Below we review the MIPv6 security. The reviewed problems and solutions are not really relevant to our work in this part of this thesis. However, our work is influenced by the lessons learned from MIPv6 security experience: we cannot assume the availability of a global security infrastructure for authenticating the packets coming from anywhere in the Internet. Therefore, we start with a short story of MIPv6 security.

In MIPv6, without any specific defense integrated into the protocol, an attacker can remotely change the binding caches held by correspondent nodes. An attacker can spoof the home address of a mobile node and send a bogus binding update to a correspondent node, where the bogus binding update points to a wrong care-of-address, the attacker's address or a victim's address. This is interpreted as an IP handover by the correspondent which, according to MIPv6, should send its packets to the mobile host's new care-of-address.

Three types of attacks are possible using the same malicious technique. In the first one the attacker can choose a random care-of-address, in which case the communication between the correspondent node and mobile host cannot even start (when initiated by the correspondent node) or stopped. In the second, the attacker can choose its own address, in which case the correspondent node's packets sent to the mobile host can be captured, regardless of the attacker's location. The attacker can play the role of an intermediate between the correspondent node and the router and hi-jack upper layer connections, obtain sensitive data etc. In the third one, the attacker can modify the binding caches of correspondent nodes that generate heavy data stream and point them to a victim node's address. The victim node will probably be flooded with unwanted data[44].

15.1.1 IPsec

IPsec (IP security) is designed to provide security for IPv4 and IPv6[45]. The security services offered by IPsec include confidentiality, and packet origin authentication. These services are pro-

vided at the IP layer, hence IP and/or upper layer protocols are protected. IPsec is composed different protocols: Authentication Header (AH)[46] for packet authentication, Encapsulating Security Payload (ESP)[47] for authentication and confidentiality, Internet Key Exchange (IKE)[48] for establishing security associations.

IPsec provides end-to-end security. However, before the communication between two peers can be secured using AH or ESP, the two peers must establish a security association using IKE. A security association consists of sharing a secret key and exchanging other control information. IKE uses public key cryptography. Each Internet node generates a public/private key pair. The public key is made known to any other node, whereas the private key is kept secret. Given the public key, it is impossible to guess the private key since this requires many years of computation. During IKE, the communicating peers exchange their public keys which can be used to generate a same secret at both ends without transmitting it. Unfortunately, without proof of public key ownership, public key cryptography is prone to man-in-the-middle attacks. An attacker on the path between the communication ends A and B, can posture as B for A and as A for B. In other words, the attacker can establish two security associations with A and B, where A and B believe that the other end of their security association is B and A respectively. The attacker can listen and modify the packets in transit between the communicating peers.

The earlier versions of MIPv6 recommended the use IPsec for securing MIPv6 against the above mentioned attacks, i.e. for authenticating and authorizing the binding updates. Later, it turn out that IPsec cannot be used this way due to the absence of a global Public Key Infrastructure (PKI) assumed by IKE. The problem is that, in MIPv6 a mobile host can communicate with a correspondent node that is located anywhere in the Internet. Hence, without a global PKI the mobile host and correspondent node cannot prove that they own their claimed public keys. The same problem exists with secret key infrastructures. They can be used locally, but they do not scale.

Fortunately, a mobile host and its home agent belong the same administrative domain and can be manually configured with a secret key. In this case IPsec can be reasonably used for securing the binding updates sent to home agents. However, this does not solve the binding update authorization problem with correspondent nodes.

15.1.2 Return Routability

The binding update authorization problem in MIPv6 was interpreted as a more general "address ownership" problem in IPv6[49]. All the mobile host needs is to prove to the correspondent node that it owns its IPv6 address, in which case the binding update authorization can be automatically granted. A mobile host is authorized to change the routing state concerning its own address.

The designers of MIPv6 security started with a very strong mechanism and ended with a weak but simpler mechanism. The initial solution was using a CGA (Cryptographically Generated Address)[50][51]. The CGA approach uses public key cryptography, but does not require an infrastructure. A mobile host generates a public/private key pair. By computing a hash of its public key, the mobile host generates its CGA. The host can send a binding update with the CGA and sign it using its private key. The binding update also carries the mobile host's public key, allowing the correspondent to node check the signature. This procedure allows the correspondent node to ensure that the address was generated using the mobile host's public key and the mobile host owns (i.e. generated) that address.

The currently adopted solution does not require public key cryptography nor security infrastructure, but it is weaker. It is called *Return Routability*[22][44]. Using the return routability procedure, the correspondent node is allowed to test whether packets addressed to the claimed home and care-of addresses are routed to the mobile node. The mobile node can pass the test only if it is able to supply proof that it received certain data called *keygen tokens* sent by the correspondent node. These are combined by the mobile node into a binding management key, denoted K_{bm}. The return routability procedure is shown in Figure 15.1. In order to obtain binding update authorization from the correspondent node, the mobile host generates two messages: *Home Test*

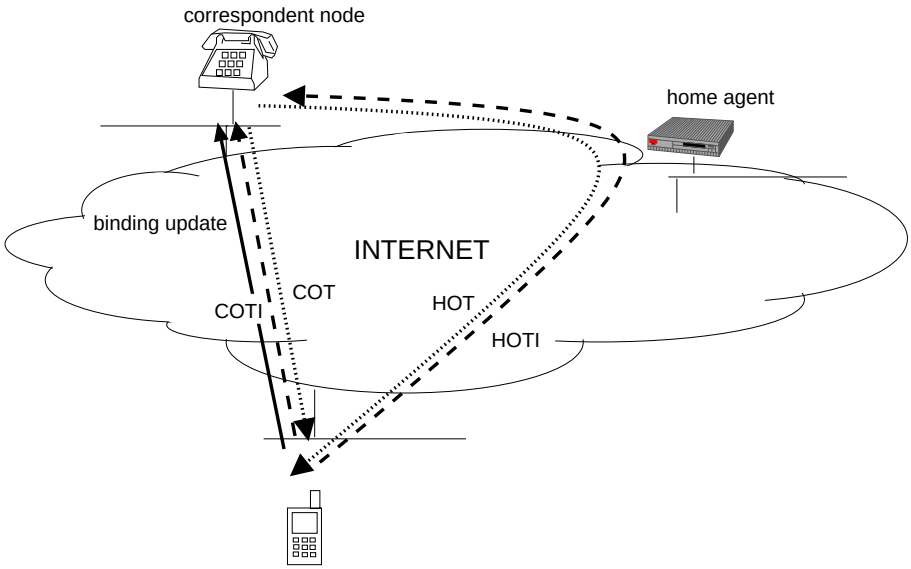


Figure 15.1: Mobile IPv6 Return Routability Procedure.

Init (HOT) and *Care-of Test Init* (COT). These messages initiate the test of the mobile host's home address and care-of-address. The HOTI message is reverse tunneled to the correspondent node through the home agent, the COTI message is directly sent to the correspondent node's IP address. The correspondent node returns two messages: *Home Test*(HOT) and *Care-of Test* (COT). The HOT message is sent to the mobile host's home address and tunneled to its care-of-address. The COT message is directly sent to the mobile host's care-of-address.

Both HOT and COT messages contain a cryptographically generated (by the correspondent node) token. The HOT message contains:

$$home\ token = hash(K_{cn}|source\ address|nonce|0)$$

where K_{cn} is a secret key only known to the correspondent node, source address is the source address of the HOTI message (i.e. mobile host's claimed home address) and 0 means HOT message. The HOT message is sent to the mobile host's home address, which is tunneled to the mobile host's care-of-address through a secure channel. In order to receive its content the mobile host must own its claimed home address and have recently sent a binding update to its home agent. Otherwise, the mobile host cannot receive the message. Similarly, the COT message contains:

$$care-of\ token = hash(K_{cn}|source\ address|nonce|1)$$

where source address if the source address of the COTI message, i.e. the mobile host's claimed care-of-address, and 1 means COT message. The COT message is directly sent to the mobile host's claimed care-of-address. Note that in order to receive its content the mobile host must be located in the subnet prefix of its claimed care-of-address.

The mobile host is required to prove that it has received the *home token* and *care-of token*. The mobile host generates the binding management key

$$K_{bm} = hash(home\ token|care-of\ token)$$

which is used for binding update authorization. The mobile host sends a binding update that contains K_{bm} to the correspondent node. Subsequent binding updates can also be authorized using the same key as long as it remains valid.

The return routability procedure provides a correspondent node with some assurance (against most of the practical attacks) that a mobile host is reachable through its claimed home address

and at its claimed care-of-address, before accepting its binding update. The procedure does not require the configuration of security associations or the existence of an authentication infrastructure between the mobile nodes and correspondent nodes. No protection is offered against bogus binding updates generated by the attackers who have access to the path [home agent ↔ correspondent node]. However, attackers in such a location are already capable of mounting similar attacks using existing security flaws even without Mobile IPv6. The main advantage of the return routability procedure is that it limits the potential attackers to those having an access to one specific path in the Internet, and avoids bogus binding updates from anywhere else in the Internet.

15.2 Problem definition

15.2.1 Existing attacks against low-end devices

MIPv6 security does not attempt to protect low-end mobile hosts from existing DOS attacks that consume energy and CPU cycles, and network bandwidth. DOS attacks against CPU and bandwidth resources are straightforward, and consist of continuously sending many bogus packets to one or more target nodes. These attacks are considered more harmful in a wireless Internet where bandwidth is scarce, and hosts are CPU and energy constrained.

15.2.2 Some bandwidth threats in MIPv6

Attacks are also possible by sending many packets to many home addresses served by one or more home agents, for editorial convenience, we call them "massive MIPv6 attacks". Since home addresses are static and publicly available, the attacker can collect a large number of home addresses. In [52][37][53], we argued that these attacks may result in remotely triggered high volume of paging i.e. broadcast traffic since mobile hosts are idle most of the time.

In addition, the home agents will consume CPU cycles for consulting binding caches and tunneling the packets. By doing so, the home agent also acts as, what we can call a "DOS repeater". Note that in IPv6, without home agent, malicious packets would be lost in the target subnet. The MIPv6 home agent generates another and larger packet (with encapsulation overhead) in response to each received malicious packet. The bandwidth cost will depend on network topology and the location of destination mobile hosts.

Another impact of massive MIPv6 attacks is DOS amplification of factor 2, by mobile hosts in routing optimization mode. In response to a packet tunneled by the home agent, each mobile host will start the return routability procedure with a correspondent node and generate two packets, HOTI and COTI.

In bidirectional tunneling mode, or routing optimization mode the home agent tunnels malicious packets that will be reflected to itself from many different destination mobile hosts. In bidirectional tunneling mode, the reflected packet is the mobile host's reply to the correspondent, which is routed through the home agent. In the routing optimization mode, the reflected packet is the HOTI message. By periodically spraying a very large number of home addresses, the attacker node make sure that the mobile hosts are "not" down (i.e. still reflect packets), while the malicious bandwidth is well exploited. In other words, many reflected packets can be generated although each destination mobile host is rarely injected a malicious packet. This attack is similar to distributed denial-of-service attacks using reflectors[54], except that the target (home agent) itself contributes to the attack by tunneling the malicious packets to their destinations. The home agent (and home network) receives high volume traffic from the attacker and also from many mobile hosts.

15.3 Proposal overview

A mobile host is not easily reachable without its home agent. An attacking node cannot easily detect the current care-of-address of a mobile host, unless it is capable of monitoring its binding

updates. As a result, we estimate that mobile hosts will be mostly attacked using their home addresses. This is a security advantage, since the home agent acts as a single point of entry where packets sent to the mobile hosts can be controlled. However, currently, MIPv6 does not profit from this advantage.

Basically, we argue that in MIPv6, packet authentication should be performed by the home agent on behalf of the destination mobile host before tunneling the packet. Strong authentication cannot possibly scale, since the home agent cannot establish IPsec security associations with all correspondent nodes. Therefore, we propose weak authentication which provides reasonable assurance that the correspondent node does not deploy IP spoofing.

The home agent can profit from a mechanism that is very similar to SYN cookies in order to authorize incoming sessions. SYN cookies will provide a home agent with some assurance that the session initiator is reachable at its claimed IP address. Attacks are possible by using topologically correct source addresses. However, the attacking nodes can be easily traced back to their networks, in this case. Because the above defined attacks cannot succeed without generating large numbers of packets. A large number of suspicious packets from the same network is a possible sign of attack.

Thus, our goal is to augment MIPv6 with some reasonable assurance that home agents do not route spoofed packets to mobile hosts, and/or attacks are discouraged by tracing them back to their origin.

Chapter 16

Home agent SYN cookies

Contents

16.1 Introduction

In this chapter we propose a technique for improving the security of incoming sessions in MIPv6, using TCP SYN cookies. TCP SYN cookies were designed several years ago with the goal of defeating SYN flooding attacks. In traditional SYN cookies approach, the weak authentication is performed by the receiver, before resources are allocated for the incoming TCP connection. We propose that the home agent returns a SYN cookie on behalf of the destination host. We use the traditional SYN cookies approach in a different way and in order to protect different resources which are bandwidth, energy and CPU.

We assume that all sessions received by mobile hosts are initiated using SIP (Session Initiation Protocol)[55]. SIP is an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences. SIP defines a SIP INVITE message used to create sessions and perform session descriptions. A SIP INVITE message may be directly received from the session initiating node, or routed through proxy serves.

SIP does have any preference for the transport layer protocol, except for large SIP INVITE messages that may need to be transported over TCP[57]. We assume all incoming sessions start

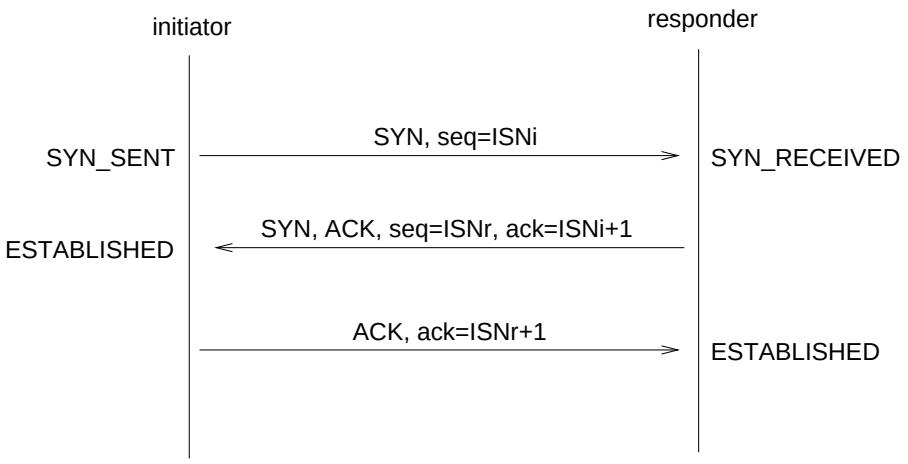


Figure 16.1: TCP 3-way handshake.

with 3-way handshake, e.g. SIP INVITE message transported over TCP. We assume that SIP over UDP[56] is either secured or dropped. In other words, we address the problem of DOS attacks with TCP packets. Other packets destined from mobile hosts, such as ICMP[9] are also dropped by home agent. TCP is the only packet type that can be sent to a mobile host (through the home agent) for starting a session.

16.2 Review of TCP SYN cookies

TCP (Transmission Control Protocol)[57] is a connection-oriented transport protocol that provides a reliable logical circuit between two communicating peers that use a less reliable IP infrastructure. In order to support reliable communication, TCP initializes and maintains certain status information for each data stream. The combination of this information including sockets, sequence numbers and window sizes, is called a connection. A pair of socket (4 tuple consisting of the connection initiator's IP address and port number, and the responder's IP address and port number) specifies the two end points that uniquely identify each TCP connection in the Internet.

For a connection to be established, two TCPs must synchronize on each other's 32-bit sequence numbers. This is achieved by exchanging connection establishing TCP packets carrying a SYN control bit and an initial sequence number (ISN). This synchronization is necessary because the communicating peers are not tied to a global clock. The synchronization requires each side to send its own ISN and receive its acknowledgment from the other side. This procedure is called a *TCP 3-way handshake* and illustrated in Figure 16.1. The first packet sets the SYN flag meaning that the sequence number is the initiator's initial sequence number ISN_i for a new connection (this packet is denoted a SYN packet). The responder returns a packet that sets the SYN and ACK flags (denoted a SYN/ACK packet). The sequence number is the responder's initial sequence number ISN_r, and the SYN packet is acknowledged by returning $ISN_i + 1$. At this point the connection enters ESTABLISHED state at initiator side, but not at the receiver side. The initiator sends the packet with an ACK flag and acknowledges the responder's initial sequence number (denoted an ACK packet).

The well-known *SYN-flooding* attack[58][59] exploits both this design and the source address spoofing possibility in IP. An attacker starts to generate SYN packets with non-existent source IP addresses and send to a target server. The SYN/ACK packets returned to the source are lost. Using non-existing source addresses ensure that the connections in SYN-RECEIVED state are not reset by a well-behaving client that sends RESET packets in response to the unrequested SYN/ACK packets. The SYN-RECEIVED state is limited to 75 seconds, in order to not to deny

service to clients with high-latency Internet access. In much less than 75 seconds, the attacker can generate a large number of bogus SYN packets until the target server's memory (allocated for TCP connections in SYN-RECEIVED state), is filled with bogus entries. As a result of this attack, legitimate clients are denied service since the server has no more resources to accepting their SYN packets.

SYN cookies[60] allow the responder to continue accepting TCP connection requests i.e. SYN packets, without creating SYN-RECEIVED state. Upon receipt of a SYN packet, the responder sets $seq = COOKIE$ and returns a SYN/ACK packet. If the SYN packet was sent by a legitimate initiator, it receives the packet and sends an ACK packet with $ack = COOKIE + 1$. The received ACK packet contains the source and destination IP addresses and port numbers which uniquely identify a TCP connection, hence the connection can be opened by the responder. I.e., the connection directly enters the ESTABLISHED state upon receipt of an ACK packet that acknowledges the COOKIE.

One information that is not available in the ACK packet is the initiator's MSS (Maximum Segment Size) value which is normally sent by the SYN packet. SYN cookies approach solves this problem by encoding an MSS value close to the initiator's MSS in the COOKIE. The MSS option is important for receiving as large TCP packets as possible without causing fragmentation. However, the MSS is a 16-bit value which if included in the COOKIE would not leave enough bits in the COOKIE for its cryptographic functions. Therefore, in order not to always use the same MSS value, the responder maintains a table of most commonly used MSS values. A 3-bit index to the largest value in the table that is not greater than the initiator's MSS is encoded in the COOKIE.

The COOKIE is constructed using the following formula [1]:

$$COOKIE = hash1 + ISN_i + count << 24 + ((hash2 + mssindex)\&((1 << 24) - 1))$$

where $hash1$ is the hash of the source port, source IP, destination port, destination IP and a secret value. The *count* value is the current value of a 32 bit minute counter. The $hash2$ value is a hash of count, source port, source IP, destination port, destination IP and another secret value. The $mssindex$ is an approximation of the MSS value found in the SYN packet.

We omit the implementation details of COOKIE verification upon receipt of an ACK packet. However, it is important to note that the responder is able to retrieve the *count* value which, when subtracted from the current minute count, gives the time in minutes since the COOKIE was created. This allows the responder to drop ACK packets with expired cookies. An attacker that is not able to monitor SYN/ACK packets of the responder must guess the 32-bit cookie. By rejecting expired cookies the responder ensures that the attacker is not given enough time for cookie prediction.

Unlike the MSS, the Window field of the TCP header can be specified in each TCP packet. However, the Window Scale Factor (WSF) option, that provides a multiplier for the value Window field, is normally sent in SYN carrying packets, i.e. SYN and SYN/ACK packets. Using SYN cookies the WSF value of the initiator is lost, since it is not encoded in the COOKIE. Therefore, the usefulness of the Window field is reduced when SYN cookies are in use. Therefore SYN cookies are only activated when the responder is under SYN flooding attack, i.e. its SYN-RECEIVED queue is filled.

16.3 Home agent SYN cookies

16.3.1 Authorizing an incoming session

A mobile host (that we call *responder*) sends its home agent a binding update or another packet that contains a randomly generated nonce $nonce_r$. The nonce is cached along with the

[1] SYN cookies as implemented in Red Hat Linux 7.2, by Andi Kleen.

mobile host's mobility binding. Note that, in MIPv6 mobile hosts have security associations with their home agent, therefore $nonce_r$ is sent through a secure channel, i.e. unknown to third parties. The $nonce_r$ value is periodically changed.

When a correspondent node (that we call *initiator*) sends a SYN packet to a mobile host's home address, the home agent intercepts the packet. The received packet is denoted:

$$TCP(P_i, P_r, seq = ISN_i, SYN), src = IP_i, dest = IP_r$$

where P_i, P_r, IP_i, IP_r are initiator and responder port numbers and IP addresses (note that IP_r is the responder's home address) and ISN_i is the initiator's initial sequence number.

Upon receipt of a SYN packet, instead of tunneling it to the responder's care-of-address, the home agent returns a SYN/ACK packet to the initiator:

$$TCP(P_i, P_r, seq = ISN_r, ack = ISN_i + 1, SYN, ACK), src = IP_r, dest = IP_i$$

where

$$ISN_r = hash_{32}(nonce_r, IP_i, p_i, p_r)$$

which is called a SYN cookie, with some modifications that will be later discussed. The home agent does not buffer the SYN packet and does not create any state. Note that the home agent acts as proxy on behalf of the destination mobile host. The source IP address of the packet is the mobile host's home address. Then, the SYN/ACK packet has the exact form of a TCP packet that is expected from the responder. If the initiator does not deploy IP spoofing, it receives the SYN/ACK packet that generates the third message of the TCP 3-way handshake.

Upon receipt of a packet of the form

$$TCP(P_i, P_r, ack, ACK), src = IP_i, dest = IP_r$$

the home agent checks the $nonce_r$ value found in the binding cache of the destination home address IP_r (if any). If

$$ack - 1 = hash_{32}(nonce_r, IP_i, p_i, p_r)$$

then, the home agent tunnels the packet to the care-of-address of the destination mobile host IP_r. Note that the receipt of an ACK packet that acknowledges the cookie, provides the home agent with some assurance that the initiator has received a SYN/ACK packet from the home agent in the recent past. Therefore, the initiator is addressable at its claimed address, which is the authorization rule.

The authorized correspondent node may immediately send other packets that follow the ACK packet. In order not to drop these packets, the home agent caches the source IP address IP_i of the authorized ACK packet (in the destination mobile host's binding cache entry). As a consequence, any type of packet from the IP address IP_i is temporarily authorized during *Delta* time units. The definitive authorization of IP_i (access grant until the end of the session) will be sent by the mobile host. Note that once the address IP_i is authorized, any type of packet from that address will be allowed.

Upon receipt of the packet tunneled by the home agent, the mobile host decapsulates it and obtain the correspondent node's packet

$$TCP(p_i, p_r, ack, ACK), src = IP_i, dest = IP_r$$

then, similarly to the home agent, the mobile host compares $ack - 1$ against $hash_{32}(nonce_r, IP_i, p_i, p_r)$. If they match, the mobile host opens a connection p_i, p_r, IP_i, IP_r.

16.3.2 Authorizing subsequent packets

Recall that in bidirectional tunneling mode (used when routing optimization is not supported by the correspondent node) all packets sent by the mobile host and the correspondent node are

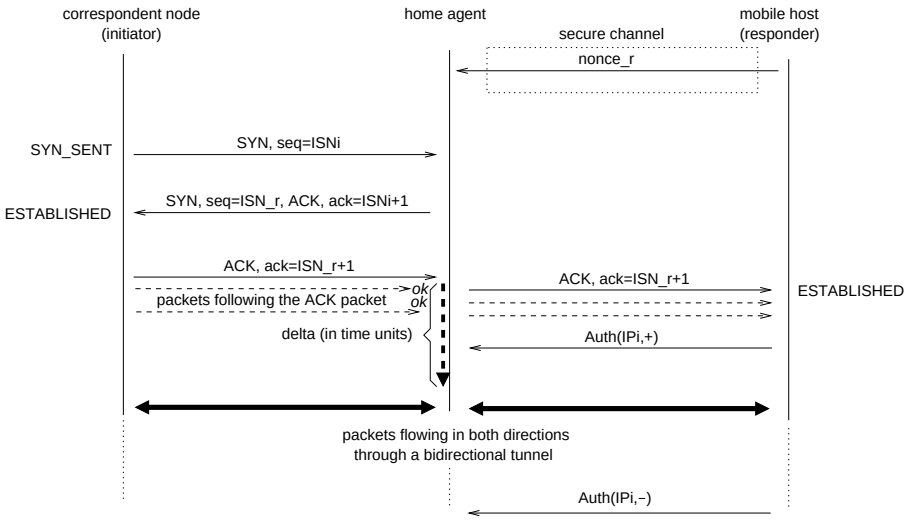


Figure 16.2: Home agent SYN cookies in bidirectional tunneling mode.

routed through the home network. Without explicit authorization from the mobile host, the home agent will accept the correspondent node's packets only during *Delta* time units after the reception of the ACK packet that acknowledges the cookie.

When the ACK packet is received and the TCP connection enters established state at mobile host side, the mobile host informs its home agent that any packet from IP_i should be allowed, i.e. tunneled to the mobile host's care-of-address. This rule is denoted $authz(IP_i, +)$ where "+" means that address IP_i is allowed send packet to the mobile host. Upon receipt of this information the home agent records IP_i the in the mobile host's binding cache. This information may be sent in separate a binding update packet to the home agent or piggybacked in a packet that is reverse tunneled to the correspondent node.

When the session is closed, the mobile host sends a $authz(IP_i, -)$ rule to its home agent, where "-" means that the address IP_i is no longer allowed. When this rule is received the home agent removes the address IP_i from the mobile host's binding cache. The IP_i can re-initiate a new session with the mobile host in the future, but it must pass (again) the weak authentication test.

In routing optimization mode, upon receipt of a valid ACK packet, the mobile host needs to initiate the return routability procedure. This requires that a HOTI message is reverse tunneled to the correspondent node and a COTI message is sent (directly) to the correspondent node. The mobile host, then, needs to receive HOT and COT messages from the correspondent node. The HOT message is sent through the home agent, the COT message is directly sent to the mobile hosts care-of-address. The HOT message will be authorized by the home agent since any packet from IP_i is authorized during *Delta* time units as described above. The *Delta* period should be long enough to allow the HOT message, without opening a long vulnerability window during which packets that spoof IP_i can be injected to the mobile host through its home agent.

In routing optimization mode, the home agent does not need to allow packets from IP_i (except the HOT message), since they are not routed through the home agent. Therefore, in routing optimization mode, the mobile host does not send a $authz(IP_i, +)$ rule to its home agent.

16.3.3 Outbound sessions

Outbound sessions that are initiated by the mobile hosts require some explicit access authorization by the home agent. Let IP_x the IP address of the correspondent node. In bidirectional tunneling mode, the mobile node can authorize the correspondent node's packets by sending a $authz(IP_x, +)$ rule to its home agent, and remove that authorization by sending $authz(IP_x, -)$ at

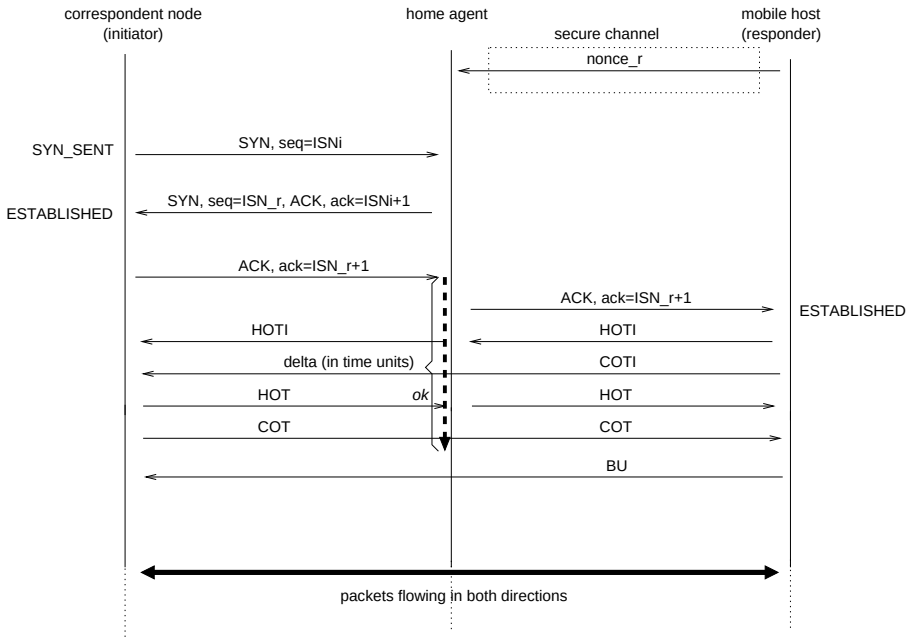


Figure 16.3: Home agent SYN cookies in routing optimization mode.

the end of the session.

In routing optimization mode, before initiating an outgoing session with IP_x the mobile host needs to complete the return routability procedure with IP_x. As a consequence, a HOT message from IP_x should be allowed by the home agent. The mobile host can ensure that the HOT message will be received by sending a $authz(IP_x, +)$ rule to its home agent. Later during the session or when the session is closed the mobile host should send a $authz(IP_x, -)$ to its home agent.

16.3.4 Congestion and network failure: late packets

Upon congestion and network failure, the communication between the mobile host and correspondent node IP_x may be interrupted and the mobile hosts binding held by the correspondent node by become stale. As a consequence, upon recovery from error or congestion, the correspondent node may need to send a packet to the mobile host's home address. This is already possible in bidirectional tunneling mode. In routing optimization mode, the correspondent node's packet can be received if the mobile host did not sent a $authz(IP_x, -)$ to its home agent. Otherwise, upon communication failure, $authz(IP_x, +)$ has to be sent the home agent to grant access to an expected late packet from IP_x.

16.4 Attack traceback

SYN cookies does not offer any protection if the attacker uses a topologically correct source address. A topologically correct source address has the attacker's real subnet prefix and allows the attacker to see the returned SYN cookie. Then, the attacker can send the right ACK packet.

However, an efficient attack requires that many malicious packets are sent and all of them have topologically correct source addresses (i.e. with the same subnet prefix). In this case, the attack can be easily detected by an administrator, the attacker's location can be traced back and administrative counter-measures can be taken. In the following discussion, we adapt this technique to MIPv6 with home agent SYN cookies.

16.4.1 Bidirectional tunneling mode

Let IP_m the attacker's topologically correct IP address. In bidirectional tunneling mode, upon receipt of a malicious ACK packet, the mobile host will send a $authz(IP_m, +)$ which grants access to packets from IP_m. Thus, subsequent malicious packets can be injected through the home address.

Since the attacker is forced to use a topologically correct source IP address, if packets are logged by the home agent administrators, an attack and its origin can be detected. In bidirectional tunneling mode, all malicious packets will be routed through the home agent. Upon detecting the attack, the mobile host can stop it by sending a $authz(IP_m, -)$ rule to its home agent. At this point the mobile user can report the attack to the home network administrator. A high rate of logged suspicious packets from the same network, sent to the same mobile host, will be a possible confirmation of attack. In this case administrative counter measures can be taken as mentioned above. The attack's origin can be blacklisted and/or its administrators can be informed.

16.4.2 Routing optimization mode

In routing optimization mode, the attacker IP_m can also send a SYN packet with a topologically correct source address to the mobile host's home address, and learn the cookie in the returned SYN/ACK packet. Then the attacker can inject an ACK packet to the mobile host, that is tunneled by its home agent. In this case, however, the mobile host will reveal its care-of-address to the attacker by sending a COTI message. Once the COTI message is received, the attacker does not have to use its real IP address. Subsequent malicious packets can have bogus IP addresses and can be directly sent to the target care-of-address.

Thus, in routing optimization mode, the home network administration cannot easily detect the attacks destined for mobile hosts. Every tunneled packet to a mobile host can be logged. However, as mentioned above, only the first malicious packet of the attack is routed through the home agent. One packet cannot possibly represent enough information to conclude that it is malicious. Therefore, we propose a simple host-based defense that protects the mobile hosts to some extent, and forces the attackers to more frequently pass through the home agent, facilitating attack origin detection by home administrators.

We propose that, upon detecting an attack or very much degraded performance, a mobile host changes its care-of-address (and also the link layer address if possible) and sends a binding update with the new care-of-address. The binding update also sends a $authz(IP_m, -)$ rule and a new nonce to the home agent. We assume that some CPU cycles can be reserved to execute the required instructions. When the care-of-address is changed, malicious packets can be quickly dropped by the target's IP module, which reduces the attack's effectiveness. The malicious packets still reach the mobile host's IP module, but energy and CPU cycles are not consumed for processing the packet content and generating replies. As a result, the attacker is forced to restart by sending a SYN packet to the mobile host's home address, with a topologically correct source IP address. However, the mobile host can change its care-of-address several times, until the attacker if forced to send enough packets to its home address, which are logged by the home administration. Each SYN packet that is logged by the home agent represents a different malicious packet stream sent to the destination. If they are received from the same network, the logs will confirm the attack, and the source network address can be blacklisted by the home administration until the attack's origin is cleaned.

An alternative solution may be delaying the routing optimization. Upon receipt of an incoming session, a mobile host can start with bidirectional tunneling mode, hence hide its care-of-address from the session initiating correspondent node, until the session is established. This may provide the mobile host with assurance that the correspondent node behaves correctly before initiating the return routability procedure which will reveal its care-of-address. The control packets exchanged at the beginning of the session may not require high performance and hence using a non-optimal route until the session is established may be acceptable. In this case, the attacker can be forced

to attack through the mobile host's home address and the attack's origin can be captured by the home administrator.

16.4.3 Packet logging details

Both in bidirectional tunneling and routing optimization mode, reducing the memory consumption and the processing overhead due to packet logging is important.

We propose that the home agent administrator logs first packet of each session tunneled to a mobile host. I.e., any packet:

1. with ACK flag set, SYN flag reset,
2. not explicitly authorized by a $authz()$ rule,
3. and acknowledges the correct cookie.

Let IP_s the source IP address of a packet $Pckt$ received from a correspondent node, and IP_d the destination mobile host's home address. Then, the home agent will execute the following:

```
1. check the binding cache of IPd
2. if(authz(IP_s,+))
        tunnel Pckt to the care-of-address of IPd;
3. else if(Pckt==TCP and SYN==1 and ACK==0)
        return SYN/ACK packet with COOKIE_IPd;
4. else if(Pckt==TCP and SYN==0 and ACK==1 and ack==COOKIE_IPd+1)
        tunnel Pckt to the care-of-address of IPd;
        log Pckt;
5. else drop Pckt;
```

Note that during a session the home agent does not perform the steps 3,4,5 (in routing optimization mode the home agent does not perform anything during a session). For each received packet the care-of-address of the destination mobile host must be fetched from a binding cache. In our case, the additional overhead is the cost of checking if the binding cache that contains the address IP_d (meaning that it is authorized). We estimate that the mobile hosts will communicate with a very limited number correspondent nodes at a time. As a result, the cost of the step 2, is probably negligible compared to that of consulting a binding cache.

16.5 Security analysis

16.5.1 Attacker location

Home agent SYN cookies approach is likely to succeed against easy DOS attacks that consist of flooding a target home address. However, home agent SYN cookies cannot cope with attackers having access to some specific paths such as [home agent $\leftrightarrow$ correspondent node] and [home agent $\leftrightarrow$ mobile host]. Obtaining unauthorized access to the IP infrastructure is probably difficult. As a consequence, the simplicity and reasonable security of the provided defense outweigths the discomfort caused by some unaddressed threats. We are also motivated by the fact that the design of MIPv6 return routability procedure is based on a similar philosophy. Below we discuss the threats that are not being addressed by home agent SYN cookies.

In bidirectional tunneling mode, an attacker on the path [home agent $\leftrightarrow$ correspondent node] can detect the correspondent node address IP_i which is granted access, spoof it, and attack the mobile host. The mobile host can possibly block the attack by sending a $authz(IP_i, -)$ rule to its home agent, which will however stop the communication with the legitimate correspondent node as

well. In practice, it is possible that there is an attacker in the same subnet as the correspondent node IP_i, which monitors its traffic and attacks the destination peer's home address. The communication will be stopped by the mobile host. This may be considered as an attack against the correspondent node, and administrative measures in the correspondent node's network may solve this problem. Note also that similar attacks already exist. For example, an attacker can also send a spoofed ICMP port unreachable message to the correspondent nodes, in which case their connection will be broken[61].

In bidirectional tunneling or routing optimization mode, an attacker on the path [home agent $\leftrightarrow$ mobile host] (or, in the same subnet as the mobile host) can detect the mobile host's care-of-address. The attacker can flood the care-of-address using bogus source addresses. Home agent SYN cookies are effective when at least the first malicious packet is sent through the home agent. Therefore, no protection can be expected if the attacker is able to monitor the current care-of-address of the target.

Similarly, in route optimization mode, an attacker on the path [correspondent node $\leftrightarrow$ mobile host] can detect the current care-of-address of the mobile host, hence no protection can be expected.

Except the above cases which are considered difficult in practice, home agent cookies provide reasonable protection for mobile hosts. Attacks are limited to the malicious nodes having an access to one specific path/location in the Internet.

16.5.2 Cookie security

The 32-bit cookie is constructed by the home agent, by calculating a hash of the mobile host's secret nonce, source IP address, source and destination port addresses. Note that we do not have any mechanism for detecting the SYN/ACK packets with stale cookies. This is because, the cookies that are calculated by the mobile host and its home agent must be the same, but they do not have synchronized time counters. A mobile host's cookie is changed by sending a new nonce. A new nonce can be sent along with a binding update to the home agent, periodically, upon movement, upon incoming session or attack.

One possible attack would be directly sending many valid ACK packets to the mobile host in order to consume its memory with TCP connections in ESTABLISHED state. This attack is avoided because the nonce value is sent to the home agent through a secure channel.

16.5.3 The *Delta* window

In routing optimization mode, a correspondent node IP_i address is allowed by the home agent during *Delta* time units (in bidirectional tunneling mode, it is allowed until end the of the session). The *Delta* period should be long enough in order not to drop the expected HOT message from IP_i. The remaining packets are flow through an optimal route between the mobile host and the correspondent node.

During the *Delta* period, an attacker can spoof the address IP_i and can transmit any type of packets to the mobile host. The mobile host may arrive at the wrong conclusion that IP_i is malicious and stop the communication by configuring a new care-of-address.

In order to detect that IP_i is temporarily allowed, the attacker will probably need to monitor its traffic, which is difficult in practice. Note that an attacker that is able to monitor the traffic of IP_i is not limited by the *Delta* window, since the mobile host's care-of-address will also be found in the same monitored packets. In this case, malicious packets can be sent to the care-of-address even when the *Delta* window expires.

Chapter 17

Discussion and conclusion

Contents

17.1 Summary

We have exploited the fact that a mobile host is not easily reachable without its home agent. Unless an attacker can predict or monitor a target mobile host's current care-of-address, the malicious packets need to be sent to its home address. In this case, the home agent acts as a single point of entry where malicious packets can be controlled and eventually stopped. The attacks that are being stopped are traditional attacks that overload a target with an outstanding number of packets with the goal of consuming CPU and energy resources.

We have proposed a mechanism, home agent SYN cookies, to ensure to a reasonable extent that home agents do not forward malicious packets to the served mobile hosts. We have shown that SYN cookies that were initially designed for defeating TCP SYN flooding attacks, is well adapted to our objective. Upon receipt of a packet, the home agent acts as a proxy on the first packet and returns the packet that is expected by the correspondent node, but also an unpredictable SYN cookie. Unless the next packet from the correspondent node is included the correct cookie, the correspondent node's packets are not forwarded to the destination mobile hosts. Attackers that deploy IP spoofing, cannot possibly see the returned cookie, hence defeated. Attackers having access to a specific path or location (e.g. the mobile host's or correspondent node's subnet) are able to monitor the returned cookies, i.e. not defeated.

Attacks are also possible, if the malicious packets have a topologically correct source address. For these cases, we have developed mechanisms to facilitate packet logging which can be used for tracing attackers back to their origin and stopping them through administrative means.

In summary, home agent SYN cookies serve two different goals in the process of protecting a mobile host from DOS attacks. First, packets with spoofed addresses are dropped before they reach the mobile host, which saves bandwidth, CPU and energy. However, this does not prevent attackers that use topologically correct source addresses. Second, malicious packets with topologically correct addresses are logged by the home agent administration instead of mobile host. The second item is important, because detecting the attack origin has no use without administrative

power. Home agent administration has the possibility of taking administrative counter-measures such as informing the other administrators. An individual user has not such a power. The user may not even be able to convince a foreign administrator, that the attack occurred.

17.2 HMIPv6 notes

In routing optimization mode, only the first malicious packet is routed through the home agent (if they have a topologically correct source address) where it is logged. Upon receipt of the first malicious packet the mobile host reports its current care-of-address to the attacker. Subsequent malicious packets can be directly sent to the mobile host. Therefore, we have defined some host-based counter-measures to force the attacker to restart the attack, allowing the home agent to log the attack source again. An attacked mobile host changes its care-of-address and sends a binding update to its home agent. En effective attack requires that another packet is sent to the mobile host's home address, otherwise malicious packets sent to the old care-of-address will not reach the victim's IP module.

In HMIPv6, the MAP (Mobility Anchor Point) also acts as a single point entry where incoming packets can be controlled. Essentially, the MAP is a local home agent (in a foreign visited network) and can deploy the same SYN-cookies approach to protect the mobile hosts in its domain. This is especially attractive in routing optimization mode, since all malicious packets must be routed through the MAP where they can be logged.

17.3 Transparency

It is important to note that there are more secure weak authentication mechanisms that we could deploy such as client puzzles[62]. In client puzzles, the initiator is returned a puzzle in stead of a cookie. The initiator must solve the puzzle and resend its request with the solution. If the solution is correct, the responder processes the initiator's request. The key idea is that the verification of a puzzle solution is inexpensive, while the difficulty of the puzzle can be increased. In case of attack, the attacking node (that floods the target with requests) needs to use a topologically correct source addresses in order to receive the puzzles, and also consume computing power for solving the returned puzzles. As a result, the attacker is forced to make much more effort than the receiver.

We have not proposed the above mechanism because it requires that every correspondent node in the Internet implements it. MIPv6 was designed to work with currently available implementations as well. In bidirectional tunneling mode, a correspondent node that is not aware of MIPv6 can transparently communicate with a mobile host. The routing optimization mode is more efficient but requires support from correspondent nodes, hence optional.

Thus, we proposed a mechanism which is transparent to current TCP implementations. Internet telephone calls, multimedia distribution, and multimedia conferences with mobile hosts can be established using SIP, and SIP INVITE messages can be transported over TCP. In this case, we can reasonably assume that every incoming session will start with a SYN packet and protected using home agent SYN cookies.

Unfortunately, SIP invite messages may be transported using UDP as well. As a result, unless SIP over UDP has its own DOS protection, our proposal is only semi-transparent. We do not change the correspondent node implementations, but we require that they transmit they SIP INVITE messages using TCP (which they implement already).

17.4 TCP performance issues

We have not encoded the session initiator's MSS (Maximum Segment Size) and WSF (Window Scale Factor) values into the home agent cookies. As a result, default values will be used by the

mobile host. Other solutions may be possible.

Note that when incoming TCP is used for transporting SIP INVITE messages, that TCP connection will be short-lived, hence the performance drawback of sub-optimal MSS and WSF values will be small. TCP performance is more important for file and e-mail transfer, and web transactions. By current practice, mobile host's do not run public FTP[63], SMTP[64] or HTTP[65] servers. FTP and Web transactions are initiated by mobile hosts. Mobile hosts periodically check e-mail using POP[66] or IMAP[67]. However, this is the current practice. Exceptions may occur, in which case the home agent SYN cookies may need an extension for using MSS and WSF values that are chosen by the correspondent nodes.

17.5 Comparison to IP traceback

Although we have followed a different path, we arrived at the a similar point as "IP traceback" mechanisms that are proposed for locating the origin of DOS attacks. Extensive research has been done in this area. IP traceback, however, is not a mechanism that forces attackers to use topologically correct source addresses. Instead they propose extensions to the routing infrastructure to learn the real origin of packets with spoofed source addresses. Below we review the some of the IP traceback proposals.

The IETF working group ITRACE has been developing a mechanism called an ICMP Traceback message, which must be supported by every router in the Internet[68]. When forwarding packets, routers generate a traceback message that is sent along to the destination. With enough Traceback messages from enough routers along the path, the traffic source and path of spoofed packets can be determined. The probability of traceback message generation is very low, e.g. about 1/20000. Assuming the average maximum diameter of the Internet is 20 hops, the resulting overhead of traceback message .1% during normal operation. In case of DOS attack, a high volume traffic from the attacking node will probably induce enough traceback messages to construct the path followed by the malicious packets. An important problem is the authentication of traceback messages. An attacker may try to generate fake traceback messages, in order to hide the attack's origin.

A different approach is hash-based traceback[69]. In this scheme, routers log every single packet, which also allows tracing low-bandwidth DOS attacks. In order to reduce the storage requirements, the authors propose packet digesting using Bloom filters. Router store a very compact representation of recent packet routing history.

Another scheme proposed for packet traceback relies on a special encoding of the path in the IP header's ID field[70]. A packet's ID field is changed by routers to provide the destination with information about the path. The recipient decodes this information to reconstruct the path to the source. No extra traffic is generated, which is an advantage over ICMP traceback. In addition, the trace information is inserted into the packets, and hence not blocked by firewalls. Unfortunately, the ID field cannot be changed if the packet is protected using end-to-end IPsec and IPv6 has an ID field.

17.6 Related work on mobile host protection

We could find one reference that we can cite, concerning mobile hosts protection. The authors do not propose any solution, rather they state that mobile host protection against DOS in a public network, is an open issue[71], because they cannot be protected using traditional firewalls. A mobile host is like a public WWW server, i.e. it expects incoming sessions from any Internet node. Thus, packet filtering does easily not apply.

The absence of previous work is probably due to IPsec. IPsec offers DOS protection by quickly rejecting malicious packets[72]. However, it is assumed that the protected mobile host has a previously established security association with the session initiating correspondent node, which is not realistic. Alternatively, the cellular domain could be protected using an IPsec security gateway.

In this case also, the security gateway must have security associations with other security gateways in the Internet. Lessons learned from MIPv6 security (in the area of binding update authorization) have shown that such assumptions are not realistic, since a they require a global PKI. A global PKI will probably never exist, according to [73]. Currently IPsec is used between a limited number of security gateways with manually configured security associations; a configuration called VPN (Virtual Private Networks).

Chapter 18

Thesis conclusion

Contents

18.1 Bloom filters, IPv6 and hash-based broadcast queries

In this thesis, our work was focused on IPv6[2]. Most of our work consisted of looking for an answer to the following question: Is there an optimization that is specific to IPv6? We could find one that consists of representing multiple 128-bit IPv6 addresses using a single 128-bit Bloom filter. Bloom filters were invented by Burton Bloom in 1970 [10], and have found other interesting applications in the networking area[13]. Bloom filters are well-adapted to IPv6 because IPv6 addresses are large. A 128-bit Bloom filter can be inserted 9 different elements at a small risk of false positive which is roughly $\frac{1}{1000}$. A false positive occurs when the membership test of an element gives a positive result although the element was not explicitly inserted to the Bloom filter.

We have used the above mechanism for an optimization, which is itself generic. We can call it *hash-based broadcast queries*. In the networking area, "broadcast" is a very common technique and generally used for sending a query to a host, when the network does not have exact routing information for that host. The query in general is sent in order to reestablish the connectivity with the destination host. The broadcast query carries the address (identifier) of the subject host. If the destination host is reachable through broadcast, it responds by returning the necessary information and *a more bandwidth efficient* connection is obtained between the sender and the responder: the subsequent packets do not have to be broadcasted. Example practices of this procedure are:

- Address resolution in IP networks,
- Paging in cellular systems,
- Route discovery in mobile ad-hoc networks (MANET).

Address resolution is used in IPv4 and IPv6 in order to discover the link layer address of a host in the same subnet, so that packets can be sent. In cellular systems such as GSM[28], paging consists of locating a dormant terminal in its paging area so that an incoming call can be delivered. In MANET, route discovery is used by a session initiator for establishing, or learning a bidirectional route with a destination host. This procedure is defined in [74] and also used in some other MANET protocols.

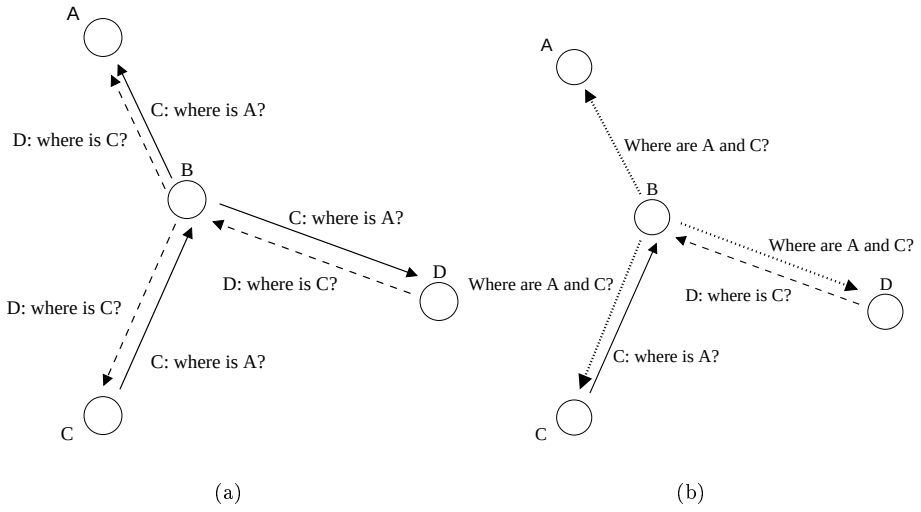


Figure 18.1: Possible application of hash-based broadcast queries to MANET (future work).

All of these procedures use the same technique which is broadcast. In address resolution, an address resolution request is broadcasted over the subnet. In cellular systems, a paging message is broadcasted over a paging area. In MANET, a route request message is broadcasted over the ad-hoc network. In any case, a broadcast query is considered bandwidth inefficient. Sending a broadcast query to an IPv6 node has the following advantage. An IPv6 node is already represented by at least 128-bit. As a consequence, a broadcast query targeted at a IPv6 node must carry a 128-bit field, which can be effectively replaced by a Bloom filter. The same query can be broadcasted to multiple nodes and thereby bandwidth can be saved. We call this technique: a hash-based broadcast query.

In this thesis, we have adapted this technique to address resolution in IPv6 and IP paging in Mobile IPv6[22]. Whether it is suitable to MANET is still an open issue. As part of future work we will be working on the following protocol illustrated in Figure 18.1. In this example, there are 4 MANET nodes. The nodes A, C, D do not hear each other (i.e. outside the transmission range). The node B hears A, C and D. The node C needs to send a packet to A, the node D needs to send a packet to C. As a consequence, C and D initiate the route request procedures denoted *C: where is A?* and *D: where is C?*. The node B hears these requests and separately broadcast them in its wireless transmission range (Figure 18.1-a). The destination nodes A and C hear the message and send their replies (the replies are not shown). This is standard MANET.

If the route request messages of C and D arrive simultaneously, B can construct a Bloom filter that represents the destination nodes, i.e. A and C and broadcast a single query (Figure 18.1-b). A and C can detect their membership to Bloom filter and reply. This procedure can possibly be generalized. I.e. a node can receive two hash-based route requests, combine them into a single Bloom filter (by computing a bitwise-OR of two Bloom filters) and broadcast one route request.

18.2 Adaptive IP paging architecture

As part of an earlier work, we have been working on an adaptive IP paging architecture. Adaptive IP paging has been previously defined in [39]. In adaptive IP paging, mobile hosts have different and time-varying paging area sizes that are adapted to their personal mobility and incoming session arrival rates.

Adaptive paging requires variable sized paging areas. In this thesis we have proposed one

architecture and algorithm for addressing this problem. In the proposed architecture, a paging area is represented as data, i.e. a list of subnet prefixes.

- The network, with some help from mobile hosts, configures variable sized paging areas that area designed for the aggregate.
- An idle mobile host, computes its optimal paging area size that depends on its incoming session arrival and mobility rate, and downloads a paging area that has the optimal size.

Our work mainly focused on the first item. The "network aggregated paging area shapes" approach is mainly motivated by scalability concerns. The network cannot possibly configure, nor store, a different paging area shape for each host. Thus we have defined a collaborative paging area configuration algorithm. Upon movement, with a very small probability e.g. $p = 1\%$ a mobile host sends a sample to a regional paging area configuration agent. A sample is a randomly chosen couple of adjacent subnets that the mobile host has visited, possibly in the recent past. The identity of the mobile host that generated the sample is not taken into account. Mobile hosts that move in the same domain send samples to the same agent. The total signaling cost of the samples is $p = 1\%$ of the binding update signaling cost, hence negligible. Since every mobile host participate in the sampling process enough samples can be eventually collected by the paging area configuration agent. Using the samples, the paging area configuration agent computes the direction probabilities in each cell, and calculates variable sized paging areas.

A mobile host that is not in active communication downloads an optimal sized paging area (a set of subnet prefixes) from its paging agent, and enters idle mode. In idle mode, the host detects if it has left its paging area by comparing its current subnet prefix to the list of prefixes that it has downloaded. Within the paging area boundaries the host does not send binding updates, upon leaving its paging area the mobile host sends a binding update to its home agent and downloads a new paging area that is computed relatively to its current subnet.

The proposed paging area configuration mechanism is centralized. A paging area configuration agent computes all the paging areas in its domain. The computation cost does not necessarily represent an important overhead, since the paging areas do not need to be computed in real-time. Using the samples received from mobile hosts, the paging area configuration agent can configure and update the paging area shapes periodically and in its idle time. On the other hand, paging area storage requirements may be important, depending on the desired paging area granularity.

18.3 Network-based vs end-to-end IP paging

Previous efforts on IP paging focused on a network-based paging service such as the one being offered in current cellular systems. The goal of a network-based IP paging service is to allow idle mobile hosts to conserve energy within a paging area. A paging area covers a wide geographical area and within paging area boundaries mobile hosts do not send location updates and enter an efficient dormant mode in which their receiver circuits are mostly turned off.

We argued that "end-to-end IP paging" is also possible. In this case IP paging is not a network-based service. End-nodes (mobile hosts and correspondent nodes) collaborate to counter the unwanted impacts of a possible home agent failure. We have replaced the paging area concept by a *living area*. A mobile host that rarely leaves its living area is likely to be located using end-to-end IP paging, by a critical correspondent node that was explicitely sent the living area information. The correspondent node can -itself- page the mobile host in its living area, in case of home agent failure. End-to-end IP paging is useful because it is likely to succeed in most cases, i.e. for mobile hosts that rarely leave their living area.

We have defined an end-to-end IP paging protocol that does not need any modification to the network. Although a minor modification to the access router code is desirable, the protocol that we have defined can work without network support. A mobile host configures a set of subnet prefixes that covers its living area, a unique 64-bit interface identifier and sends them to its critical

correspondent nodes. Upon movement within the living area, the mobile host does not need to send binding updates to its critical correspondent nodes. A critical correspondent node that wishes to communicate with the mobile host can send the copies of its session initiating packet to the possible care-of-addresses of the mobile host (different subnet prefixes + the fixed interface identifier). An obstacle (that we consider minor) is the address collision possibility. Upon movement, the mobile host may suffer from interface identifier collision and need to change the interface identifier that it has previously registered with its critical correspondent nodes. We are not discouraged by this problem. In theory, IPv6 interface identifiers may be unique on a very large scale. Currently, many other researchers work on this problem, since address collisions also degrade the Mobile IPv6 fast handover performance.

Upon home agent failure, a critical correspondent node should not flood the Internet for locating the destination mobile host. One packet should be sent at a time. Upon receipt of a destination unreachable indication from one access router, the mobile host can be paged in another subnet of its living area. In order to increase the paging speed, we have proposed a fast negative acknowledgement (destination "probably" unreachable indication). If the mobile host does not quickly respond in a candidate subnet of its leaving area, the critical correspondent node will receive a fast negative acknowledgement from the intervening access router. Then, the correspondent node can try in another subnet. In the mean time the previous access router can continue to probe the mobile host in order to make sure that it is indeed unreachable in its subnet. If the fast negative acknowledgement was premature, the mobile host will send a late reply to its access router, and receive the correspondent node's packet. In this case, the mobile host will reply to the correspondent node, while it is being paged in another subnet. If the mobile host's late reply to the correspondent node is lost, the paging procedure must be unnecessarily repeated by the correspondent node. While packet loss may occur, this is rather an exception. As a consequence, we have prefered increasing the end-to-end paging speed by assuming good network performance, rather than unnecessarily slowing down end-to-end paging for dealing with packet loss.

18.4 Network-assisted SYN-cookies

Unlike current cellular systems, in a wireless Internet, denial-of-service attacks that consume bandwidth and energy are easy. The cellular network opens its doors to the Internet where powerful and malicious nodes reside. In wireless networking, bandwidth and energy optimization are crucial. As a consequence, we expect that denial-of-service attacks will be one of the greatest obstacles on the way towards a wireless Internet. Unless this problem is solved to some reasonable extent, current cellular operators may not be willing to open the doors to the sessions coming from the Internet

With these considerations in mind, we have proposed using the TCP SYN-cookies approach for early weak authentication in Mobile IPv6. SYN-cookies have been used as a defense against SYN-flooding attacks that target public WWW servers. We are motivated by the fact that a mobile host is not easily reachable (to attackers) without the service provided by the home agent. Session initiation with a destination host will generally require that at least the first packet of the session is sent to its home address. Hence, the home agent acts as a single point of entry, where incoming sessions can be controlled. We have proposed that upon incoming TCP (destined for a mobile host), the home agent acts as a proxy to the first packet and returns a SYN-cookie on behalf of the destination mobile host. The procedure is transparent to the initiator's TCP, which then sends the third packet of the TCP 3-way handshake. If the packet acknowledges the correct cookie, then the home agent has reasonable assurance that the sender does not deploy IP spoofing. The home agent, then, forwards the packet to the mobile host.

This procedure improves the network signaling and mobile host security. However, it is assumed that every incoming session starts with a SIP INVITE message transported using TCP. Unless SIP over UDP has its own security mechanisms, the proposed defense is only semi-transparent.

Chapter 19

Résumé

This chapter contains a summary of this thesis in french.

19.1 Introduction

Dans cette thèse nous développons des algorithmes d'optimisation afin de réduire le coût de la signalisation IPv6 et la consommation de l'énergie due à la gestion de la mobilité dans un Internet sans fils.

Dans la gestion de la mobilité IPv6, la solution qui à été adoptée pour le routage des paquets vers des destinataires mobiles, consiste à déployer un réseau mère pour chaque mobile. Chaque mobile dispose d'une adresse fixe dans son réseau mère, connue par tous ses correspondants potentiels. Lorsqu'un hôte mobile se déplace vers un nouveau réseau IP, il configure une deuxième adresse IP ou le hôte est temporairement joignable. Afin de recevoir des paquets, le hôte mobile doit tenir au courant un routeur nommé *agent mère* dans son réseau mère, de son adresse temporaire courante. Ce mécanisme nécessite la transmission d'un paquet de contrôle que l'on appelle une *mise à jour de localisation*. Lorsqu'un correspondant ouvre une session avec un destinataire mobile, il envoie son paquet à l'adresse mère du destinataire qui est publiquement connue. Alors, l'agent mère du mobile intercepte le paquet et le route vers l'adresse temporaire courant du mobile. Le mobile reçoit le paquet de son correspondant, et la communication de bout en bout commence. Cette procedure est illustrée dans Figure 19.2.

Dans un Internet future, accueillant des milliards de hôtes hautement mobiles, des paquets de mise à jour de localisation peuvent consommer une partie importante de la bande passante dans l'Internet. Dans cette thèse on considère deux solutions différentes afin de résoudre ce problème. Ces solutions on été proposées dans la littérature.

La première possibilité, à laquelle nous avons contribué, consiste à configurer des sous-reséaux couvrant un grand nombre de cellules, et donc une zone géographique large. Ainsi, la fréquence des mouvements IP des mobiles sera réduite. Cependant, un grand sous-réseau contiendra un grand nombre de hôtes mobiles. Ceci peut aboutir à une haute fréquence de résolution d'adresse. Résolution d'adresse consiste à diffuser une requête de résolution d'adresse dans tout le sous-reseau, au début d'un

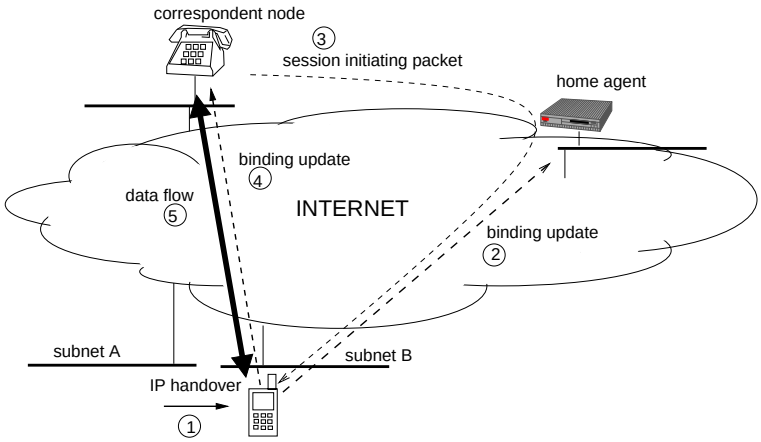


Figure 19.1: Gestion de la mobilité dans Mobile IPv6.

appel entrant. Ce problème peut poser une limitation sur la taille maximale d'un sous-réseau IP.

La deuxieme solution consiste à regrouper plusieurs sous-réseaux pour former des zones géographiques nommées des *zones de recherche IP*. Un hôte mobile qui n'est pas en cours de communication appelé *hôte dormant*, tient au courant son agent mère de sa zone de recherche courante, au lieu de sa localisation exacte. Les zones de recherche couvrent un grand nombre de sous-réseaux IP, par conséquent la fréquence des paquets de mise à jour de localisation est réduite. Cependant, au début une session entrante, un mobile doit être recherché dans sa zone de recherche courante. Ceci nécessite la diffusion d'un message de recherche dans tous les sous-réseaux de la zone afin d'informer le destinataire mobile de la session entrante. Lorsque le mobile reçoit ce message dans un des sous-réseau, il informe le réseau de sa localisation exacte, et la communication entre le mobile et son correspondant peut commencer. Cette solution réduit la fréquence des paquets de mise à jour de localisation dans l'Internet. Cependant, elle introduit un surcoût de signalisation dans les liens sans fils des sous- réseaux d'accès. La fréquence de la diffusion des messages de recherche peut devenir très haute dans une grande zone de recherche couvrant un grand nombre de mobiles. Ceci impose une limitation sur la taille maximale d'une zone de recherche.

Le problème commun dans les deux cas décrits ci-dessus, est la croissance rapide de la fréquence de diffusion de messages dans la zone de diffusion (sous-reséau, ou zone de recherche) avec le nombre de hôtes. Ceci peut facilement aboutir à une consommation importante de la bande passante sans fil dans les réseaux d'acces.

19.2 Résolution d'adresse IPv6 basé sur les filtres de Bloom

Dans cette partie de cette thèse nous développons une extension à la méthode classique de résolution d'adresse dans IPv6. Nous allons d'abord réviser IPv6 et la résolution d'adresse dans IPv6, ensuite les filtres de Bloom qui constituent le cœur de notre proposition.

Notre proposition consiste à diffuser une requête à plusieurs destinataires IPv6. En utilisant un filtre de Bloom de 128 bits, nous représentons plusieurs adresses IPv6 au lieu d'une seule. A la réception de la requête, les récepteurs peuvent facilement détecter si le message leur est destiné et répondre simultanément à la requête. Cette procédure est générale et peut probablement trouver diverses applications. Dans cette partie de la thèse nous l'appliquerons à la résolution d'adresse dans IPv6. Dans le cadre de la résolution d'adresse, la requête qui sera optimisée est la requête de résolution d'adresse. Par conséquent plusieurs adresses peuvent être résolues, en diffusant un seul message de taille standard.

19.2.1 IPv6 et résolution d'adresse IPv6

IPv6 est le nouveau protocole Internet conçu dans le but de se libérer de la limitation de plage d'adresse imposée par le protocole IP courant, c'est-à-dire IPv4. Une adresse IPv6 à une longueur de 128 bits, ce qui est considéré suffisant jusqu'à la fin des temps. Les concepteurs de l'IPv6 ne se sont pas limités à une plus grande plage d'adresses. IPv6 a été également amélioré en prennent en compte plusieurs années d'expériences obtenues dans l'Internet basé sur IPv4. Par conséquent, IPv6 améliore IPv4 en ajoutant plus de niveau de hiérarchie dans la structure d'adressage, auto configuration des adresses, une entête IP plus simple et efficace, ajout des options de IP plus flexible, et la qualité de service.

Dans IPv6 des nœuds qui sont attachés à un même lien utilisent un protocole nommé *découverte de voisin*, pour découvrir l'adresse du protocole de niveau lien. Ce processus et connu sous le nom de résolution d'adresse. L'émetteur diffuse un message de *sollicitation de voisin* dans le sous réseau afin de résoudre l'adresse du destinataire. Ce paquet contient l'adresse IPv6 du destinataire. Puisque le paquet a été diffusé en multicast, le module IP du destinataire reçoit la sollicitation. Le destinataire, alors, répond avec son adresse de protocole du niveau lien en transmettant un message nommé *annonce de voisin*. A ce stade, l'adresse du destinataire à été résolue et enregistrée par l'émetteur dans un *cache de voisin*. L'émetteur peut directement envoyer ses paquets au destinataire.

19.2.2 Résolution d'adresse IPv6 basé sur des filtres de Bloom

Nous proposons une procédure de résolution d'adresse IPv6 plus optimale en terme de bande passante, donc plus adaptée aux environnements sans fil. Le protocole que nous développons est basé sur les filtres de Bloom. Un filtre de Bloom est une structure de donnée aléatoire qui permet de représenter plusieurs éléments d'un ensemble de façon concise.

Nous utilisons un filtre de Bloom de $m = 128$ bits afin de représenter tous les éléments d'un ensemble d'adresses IPv6

$$A = \{IP_1, IP_2, ..., IP_n\}$$

La procédure nécessite k fonctions de hachage uniformes et indépendantes $h_1(), h_2(), ..., h_k()$ ou $1 \leq h() \leq m$. Avant de commencer la procédure, tous les m champs (bits) d'un

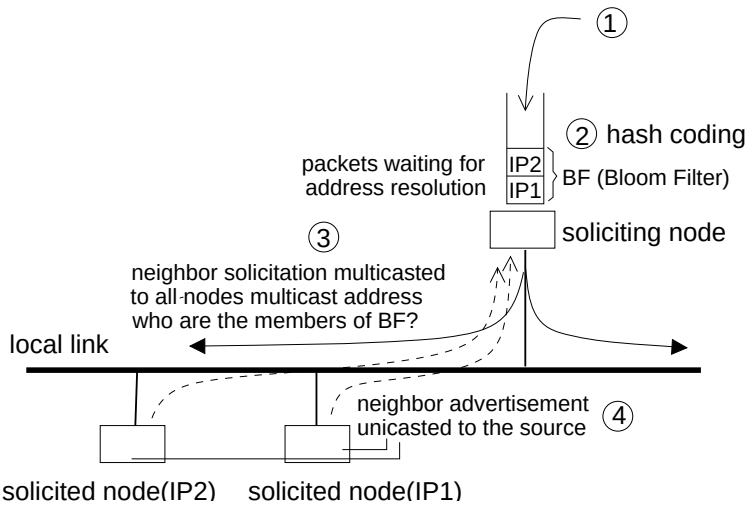


Figure 19.2: Exemple de resolution d'adresse IPv6 en utilisant un filtre de Bloom (n=2).

filtre de Bloom sont remplis de zéros. Pour chaque élément IP_i nous marquons 1 aux positions indiquées par $h_1(IP_i), h_2(IP_i), ..., h_k(IP_i)$ du filtre de Bloom. Le résultat représente tous les éléments de l'ensemble A. Afin de vérifier si une adresse IP_j est un élément de l'ensemble A, il suffit de vérifier les valeurs des bits dans les positions indiquées par $h_1(IP_j), h_2(IP_j), ..., h_k(IP_j)$. Si ils sont tous des 1, alors IP_j est un élément de l'ensemble A. Cependant, il existe une petite probabilité que tous les bits dans les positions indiquées par $h_1(IP_j), h_2(IP_j), ..., h_k(IP_j)$ soient 1 alors que IP_j n'est vraiment un élément de l'ensemble A. Cette situation est appelée un *faux positif*, et causée par le fait que des résultats de hachages des différents éléments, peuvent collisionner et indiquer la même position de bit dans un filtre de Bloom. Ce protocole est illustré dans Figure 5.5.

Le nombre optimal de fonctions de hachage k_{opt} est donné par l'équation

$$k_{opt} = (ln2) \times \frac{128}{n}$$

et en utilisant k_{opt} la probabilité de faux positif est

$$F = (0.6185)^{128/n}$$

Un nœud va représenter toutes les adresses dont il a besoin de résoudre en utilisant un filtre de Bloom, et envoyer sa requête en diffusant un seul message de sollicitation de voisin. Dans notre cas, un faux positif va générer une *fausse annonce de voisin*. C'est-à-dire, il est possible qu'un nœud mal interprète un message de sollicitation de voisin et réponde même si il n'a pas été sollicité.

Dans un sous réseau de N nœuds, en utilisant un filtre de Bloom, 1 sollicitation de voisin plus $N \times F$ messages de fausse annonce de voisin seront transmis par n adresses résolues. Le gain de signalisation dans le réseau cœur est donné par:

$$G_{coeur} = \frac{n}{1 + N \times F}$$

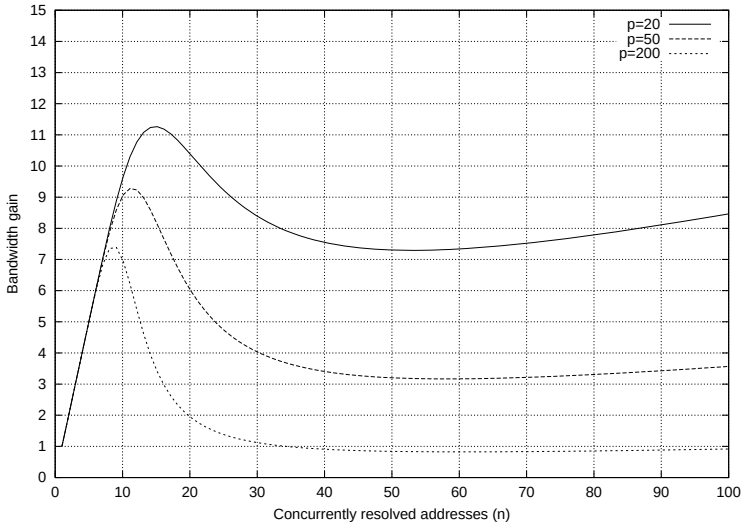


Figure 19.3: Le gain de signalisation par cellule.

On va faire l'hypothèse que le sous réseau couvre plusieurs cellules, afin d'évaluer le gain de signalisation sans fil. Nous définissons p, la densité de hôtes mobiles, c'est-à-dire le nombre de mobiles par cellule. En utilisant un filtre de Bloom, 1 sollicitation de voisin plus $p \times F$ messages de fausse annonce de voisin sont transmis dans chaque cellule, au lieu de n messages de sollicitation de voisin. Le gain de signalisation est donné par:

$$G_{sansfil} = \frac{n}{1 + p \times F}$$

et illustré dans Figure 19.3.

Il est important de noter que cet algorithme peut causer une perte de signalisation dans le cœur réseau mais un gain de signalisation sans fil. Par exemple, avec $n = 9$ nous avons $F = 0.001078$. Cela signifie un gain de $G_{coeur} = 1.4$ dans un sous réseau contenant $N = 5.000$ hôtes, et $G_{coeur} = 0.76$ dans un sous réseau contenant $N = 10.000$ hôtes. Cependant, dans les deux cas nous avons le même gain $G_{sansfil} = 8.54$ dans une cellule contenant $p = 20$ hôtes.

19.2.3 Interruptions du module IP

Le message de sollicitation de voisin standard est diffusé vers une adresse multicast calculée en fonction de l'adresse que l'on veut résoudre. Le nœud destinataire ne reçoit pas les messages qui sollicitent les adresses des autres nœuds. Ceci évite les nœuds de recevoir une interruption à chaque fois qu'une adresse est résolue dans le réseau.

Notre proposition souffre d'un problème d'interruption du module IP puisque le message de sollicitation de voisin est destiné à plusieurs nœuds. Les destinataires peuvent être abonné aux différents groups de multicast. Par conséquent, nous ne pouvons pas utiliser les adresses multicast.

Afin d'éviter une haute fréquence d'interruptions sur les nœuds, résolution d'adresse basé sur les fitres de Bloom doit être contrôlé par un seuil et nous choisissons un seuil $Se = 9$. A moins que le nombre d'adresses atteint le seuil 9, la résolution d'adresse standard est utilisée. Ceci nous permet de réduire la fréquence des interruptions. Le choix du seuil $Se = 9$ est fait en prennent en considération la probabilité de false positives. Avec $n = 9$ nous avons $F = 0.001078$ qui est acceptable: en moyenne 1 faux positif par 1000 résolutions d'adresse.

19.2.4 Résultats de simulation

Nous avons évalué la performance des filtres de Bloom dans la résolution d'adresse IPv6 dans un sous réseau comprenant plusieurs hôtes mobiles. Le routeur d'accès doit résoudre et maintenir dans son cache plusieurs adresses afin de pouvoir fournir un accès Internet.

Nous avons commencé par une simulation de 5.000 mobiles. Afin de profiter du gain de signalisation de notre algorithme, nous avons limité le taux des messages de résolution d'adresse à 1 sollicitation de voisin par second. Cela a permis au routeur d'accès de résoudre plusieurs adresses qui sont accumulées dans une file d'attente, sans introduire un grand délai de résolution d'adresse. Nous avons configuré $Se = 9$ ce qui nous a permis de réduire la fréquence des interruptions du niveau IP. Cependant, puisque le taux de résolution d'adresse est limité nous avons observé des délais d'attente. Il est important de noter que notre proposition a un cout de calcul des fonctions de hachage. Ce cout va être payé à chaque fois que le seuil Se est atteint, et le routeur va calculer $Se \times k_{opt} = 9 \times 9 = 81$ fonctions de hachage.

Nous avons observé que les délais d'attente dépendent du taux des sessions entrantes et la durée de vie des caches. Quand le taux des sessions entrantes est fixe, ou $\frac{\triangle I}{\triangle t} \simeq 0$, une longue durée de vie de cache réduit la fréquence de résolution d'adresse (ou les délais d'attente), puisque la destination d'une session, en générale, se trouve déjà dans le cache. Cependant, quand le taux des sessions entrantes augment, $\frac{\triangle I}{\triangle t} > 0$) une longue durée de vie de cache est moins utile pour réduire la fréquence de la résolution d'adresse. Dans ce cas, nous avons observé de longs délais d'attentes avec le mécanisme standard. On a réduit les délais d'attentes avec notre mécanisme, en revanche notre mécanisme a causé des interruptions dans tous les nœuds dans le sous-réseau.

Dans le pire des cas le routeur d'accès peut perdre son cache. Nous avons simulé cette situation avec 5.000 et 10.000 mobiles par sous réseau. Avec 10.000 mobiles, le mécanisme de résolution d'adresse standard a échoué puisque toutes sessions entrantes nécessitaient la résolution de l'adresse destinataire. Dans le même cas, notre mécanisme a réussit à retenir les délais en dessous des limites raisonnables, mais nous avons observé des interruptions fréquentes dans tous les mobiles.

19.2.5 Discussion

Dans la littérature nous avons noté que des fréquences d'interruption beaucoup plus élevées que celle causée par notre proposition, ont été permises. Par exemple

la spécification de Mobile IPv6 permet aux routeurs d'accès de générer jusqu'à 40 *annonce de routeur* par second pour accélérer la détection des mouvements. Ces messages génèrent également des interruptions dans tous les nœuds, donc Mobile IPv6 permet 40 interruptions par seconde dans tous les mobiles. Dans nos simulations nous avons observé une fréquence maximale de 1 interruption par 5 seconds (dans le cas de perte de cache et 10.000 mobiles par sous sous réseau).

19.3 Recherche de mobiles basée sur des filtres de Bloom

Dans cette partie de la thèse, nous développons un algorithme de recherche basé sur les filtres de Bloom. C'est-à-dire nous appliquons une optimisation similaire à la partie précédente mais à un différent problème. Recherche de mobiles, est un protocole qui a été propose dans la littérature afin de minimiser le coût de signalisation et la consommation d'énergie dans les réseaux cellulaire basés sur Mobile IP.

Le protocole consiste à localiser un hôte mobile dans une zone de recherche couvrant plusieurs cellules. Cette optimisation qui est utilisée par les systèmes cellulaires d'aujourd'hui (comme GSM), permet aux mobiles de économiser de l'énergie. Un terminal cellulaire qui n'est pas en cours de communication, ne met pas à jour sa localisation exacte dans sa zone de recherche. Cela évite le terminal de consommer de l'énergie pour signaler sa localisation (ou, cellule) au réseau à chaque fois qu'il fait un mouvement. Il est également possible de réduire l'énergie consommée par les circuits de réception. Pour cela, le terminal se synchronise avec les stations de base et périodiquement allume son récepteur. Un terminal dont la localisation exacte est inconnue par le système est appelé un terminal dormant.

Ce service n'est pas gratuit, car il nécessite une procédure appelée *recherche de mobile*. Lorsqu'il y a un appel entrant pour un terminal dormant, le terminal doit être cherché dans sa zone de recherche la plus récente. Cette procédure est coûteuse puisqu'il consiste a générer plusieurs copies d'un message de recherche et les diffuser dans toutes les cellules dans la zone de recherche. Lorsque le mobile reçoit le message de recherche dans une des cellules, il entre le mode actif (allume son récepteur), enregistre sa localisation exacte et la communication peut commencer.

Recherche de mobiles IP consiste à fournir le même service au niveau IP et en utilisant des entités IP. Dans ce cas, un hôte mobile est recherché un utilisant son adresse IP. En d'autres termes, le message de recherche contient l'adresse IP du mobile recherché. Le message est reçu par le module IP du mobile qui détecte si le message lui est destiné. Dans Mobile IPv6, un mobile sera représenté par son adresse IPv6 qui est 128-bit. Cela nous attire, puisque les filtres de Bloom peut être effectivement utilisé avec 128-bit. Plusieurs mobiles dormants dans une même zone de recherche, peuvent être recherché en diffusant un seul message.

19.3.1 Mobile IPv6 et recherche des mobiles IP

Mobile IPv6 est une solution de mobilité globale pour les hôtes IP mobile. En utilisant Mobile IPv6, un hôte IP peut changer son sous-réseau IP alors qu'il est en cours de communication avec un correspondant. Pour cela un hôte mobile est donné

deux adresses IP. La première est appelée une adresse mère. C'est une adresse fixe et globalement unique, dont le préfix IPv6 est le sous-réseau mère du mobile. La deuxième est appelée une adresse temporaire. A chaque fois qu'un mobile change son sous-réseau il configure une nouvelle adresse temporaire. L'adresse mère est utilisé comme identifiant fixe nécessaire pour maintenir une connexion TCP (ou autre), alors que l'adresse temporaire représente la localisation (sous-réseau) du mobile.

Mobile IPv6 introduit une nouvelle entité nommé agent mère. Un agent mère se situe dans le sous réseau mère et il est responsable de détecter les paquets envoyés à l'adresse mère d'un mobile et les retransmettre à son adresse temporaire courante. Donc, son agent mère permet à un mobile d'être joignable a n'importe quel correspondent utilisant son adresse mère qui est fixe et connue. Cependant, à chaque fois qu'un mobile change son adresse temporaire, il doit informer son agent mère. Le message envoyé par le mobile afin de mettre à jour son adresse temporaire est appelé un message de mise à jour de localisation.

Mobile IPv6 supporte deux modes de communication. Le premier mode est appelé un tunnel bidirectionnel. Dans ce cas, le correspondent n'a pas besoin de supporter Mobile IPv6. IPv6 standard, sans support de mobilité, peut être utilisé afin de communiquer avec un hôte Mobile IPv6. Pour cela les paquets échangés par le mobile et le correspondent doivent toujours passer par l'agent mère du mobile. Ce mode est considéré sous optimal puisque tous les paquets doivent passer par l'agent mère qui peut être dans certain cas situé dans un sous-réseau distant. Le second mode est appelé l'optimisation de routage est nécessite la collaboration du correspondent. Dans ce cas le correspondent accepte les messages de mise à jour de localisation envoyés par le mobile, et donc peut directement envoyer ses paquets à son adresse temporaire courante.

Recherche de mobiles IP a été proposé afin de réduire le coût de signalisation des messages de mise à jour de localisation. Cela fait h'hypothèse que les mobiles IP ne communiqueront pas la plupart du temps comme les terminaux cellulaires d'aujourd'hui. En cessant d'envoyer des messages de mise à jour de localisation dans une dans une zone de recherche, les mobiles peuvent contribuer à la réduction de signalisation globale dans l'Internet. Cette procédure peut également permettre aux mobiles d'économiser leur énergie.

La conception d'un protocole de recherche de mobiles IP est à peu près définie. On introduit une nouvelle fonction nommé *agent de recherche*, afin de fournir un service de recherche. Le premier paquet P1 d'une session entrante (destiné pour un mobile dormant) est intercepté par son agent de recherche. L'agent de recherche, diffuse un message de recherche destiné pour le mobile dans sa zone de recherche, le mobile envoie un message de mise à jour de localisation et reçoit le paquet P1. A ce stade la communication de bout en bout peut commencer entre le mobile et le correspondant.

Cependant, nous n'avons pas encore une expérience de déploiement Mobile IPv6, et pour l'instant il est difficile de répondre à la question suivante : qu'est- ce qu'une zone de recherche dans un réseau IP? Quelle est la relation entre une zone de recherche et sous-réseau? Dans cette thèse nous envisageons deux possibilités. Dans la première, les zones de recherche et sous-réseaux sont orthogonaux. Dans ce cas,

l'adresse temporaire du mobile est inconnue, donc il est recherché en utilisant son adresse mère qui est fixe. Dans la deuxième les zones de recherche et sous-réseaux sont identiques. Donc l'adresse temporaire du mobile ne change pas dans la zone de recherche. Dans ce cas un mobile peut être recherché en utilisant son adresse temporaire.

19.3.2 Recherche de mobiles basée sur les filtres de Bloom

Nous proposons une procédure de recherche qui est plus optimale en terme de bande passante. Dans cette procédure l'agent de recherche utilise un filtre de Bloom de $m = 128$ bits afin de représenter tous les éléments d'un ensemble d'adresses IPv6 $A = IP_1, IP_2, ..., IP_n$ qui ont récemment reçu un appel entrant (et qui sont dans la même zone de recherche). La procédure nécessite k fonctions de hachage uniformes et indépendantes $h_1, h_2, ..., h_k$ et $1 \leq h() \leq m$. Avant de commencer la procédure, tous les m champs (bits) d'un filtre de Bloom sont remplis de zéros. Pour chaque élément IP_i on marque 1 aux positions indiquées par $h_1(IP_i), h_2(IP_i), ..., h_k(IP_i)$ du filtre de Bloom. Le résultat représente tous les éléments de l'ensemble A.

Le filtre de Bloom est inséré dans un message de recherche qui est diffusé dans la zone de recherche. A la réception de ce message, un mobile IP_j peut détecter s'il est recherché en vérifiant les valeurs des bits dans les positions indiquées par $h_1(IP_j), h_2(IP_j), ..., h_k(IP_j)$. Si ils sont tous des 1, alors IP_j est un élément de l'ensemble A, donc le mobile est recherché et doit envoyer une mise à jour de localisation. Cependant, il existe une petite probabilité que tous les bits dans les positions indiquées par $h_1(IP_j), h_2(IP_j), ..., h_k(IP_j)$ soient 1 mais IP_j ne soit pas un élément de l'ensemble A. Dans ce cas, un mobile va envoyer un location update qui n'était pas nécessaire puisqu'il n'y avait pas d'appel entrant pour ce mobile. Nous appelons cette situation, une *fausse mise à jour de localisation.*

Le protocole de recherche de mobile basé sur les filtres de Bloom est illustré dans Figures 19.4 et 19.5.

Le nombre optimal de fonctions de hachage k_{opt} qui minimise la probabilité de fausse mise à jour de localisation, et donné par l'équation

$$k_{opt} = (ln2) \times \frac{128}{n}$$

et en utilisant k_{opt} la probabilité de fausse mise à jour de localisation est

$$F = (0.6185)^{128/n}$$

Dans une zone de recherche comprenant N nœuds, en utilisant un filtre de Bloom, 1 message de recherche plus $N \times F$ messages de fausse mise à jour de localisation seront transmis par n sessions entrantes. Le gain de signalisation dans le réseau cœur est donné par:

$$G_{coeur} = \frac{n}{1 + N \times F}$$

Une zone de recherche est constituée de plusieurs cellules. Nous définissons p, la densité de hôtes mobiles, c'est-à-dire le nombre de mobiles par cellule. En utilisant

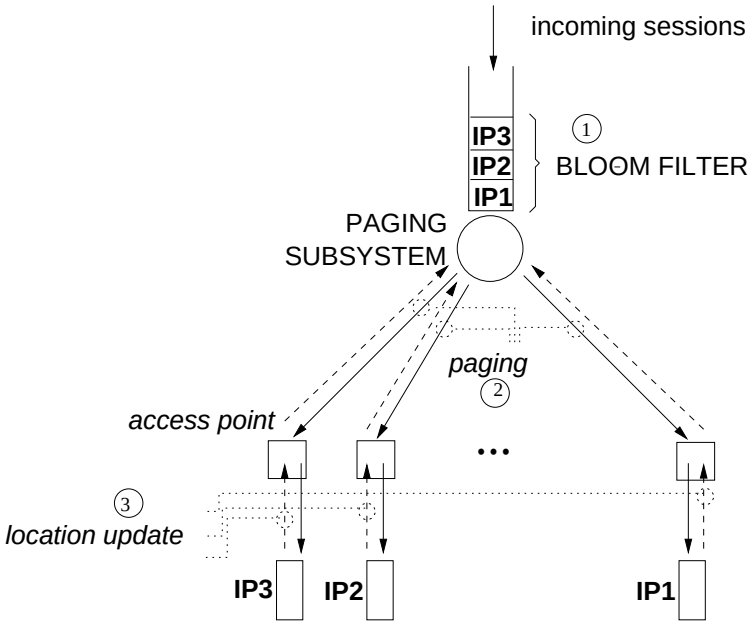


Figure 19.4: Recherche de mobile en utilisant un filtre de Bloom (une abstraction).

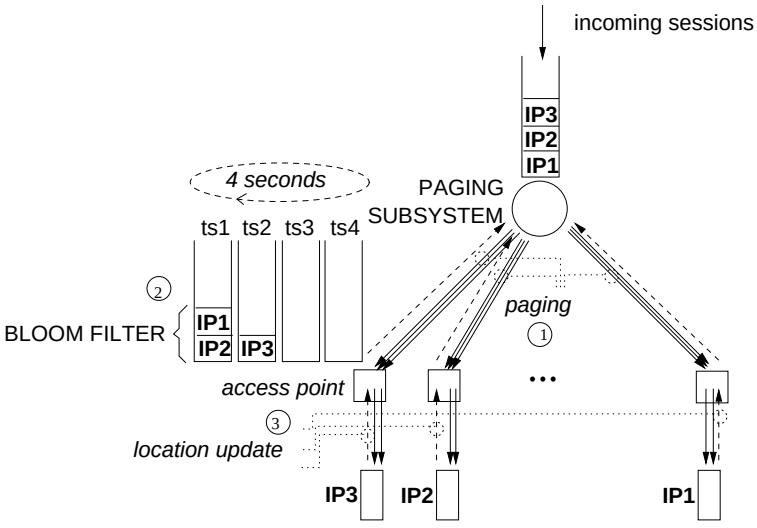


Figure 19.5: Recherche de mobile en utilisant un filtre de Bloom (appliqué par les points d'accès).

le un filtre de Bloom, 1 message de recherche plus $p \times F$ messages de fausse mise à jour de localisation sont transmis dans chaque cellule, au lieu de n messages de recherche. Le gain de signalisation est donné par:

$$G_{sansfil} = \frac{n}{1 + p \times F}$$

En mode dormant un mobile est synchronisé avec le réseau et son récepteur est périodiquement activé afin de détecter les messages de recherche. Le reste du temps, le récepteur ne consomme pas d'énergie. En faisant l'hypothèse que le récepteur est actif 1/4 du temps, le probabilité de fausse mise à jour de localisation F^{4ts} sera quatre fois plus petite que F:

$$F^{4ts} = \frac{F}{4}$$

puisque le mobile ne reçoit qu'un quart des messages de recherche.

Quand le support de mode dormant est disponible, le time-slot du mobile est déterminé par une station de base (point d'accès), en fonction de son adresse IP. Cela nécessite que notre algorithme soit appliqué par les stations de base. Dans ce cas, chaque appel entrant va générer un message de recherche reçu par chaque station de base. Cependant, au lieu de transmettre n messages de recherche dans sa cellule, chaque station de base va représenter les n adresses IP avec un filtre de Bloom et va transmettre un seul message de recherche. Cela va générer $p \times F^{4ts}$ fausse mise à jour de localisation. Il est important de noter que les stations de base peuvent filtrer les messages de fausse mise à jour de localisation. Dans ce cas, le gain de signalisation dans le réseau cœur est donné par:

$$G^{4ts}_{coeur} = 1$$

et le gain de signalisation dans chaque cellule est:

$$G^{4ts}_{sansfil} = \frac{n}{1 + p \times F^{4ts}}$$

19.4 Une architecture pour recherche de mobile adaptative

19.4.1 Recherche de mobile adaptative

Recherche de mobile adaptative améliore l'algorithme de recherche basique. Au lieu configurer des zones de recherche de taille fixe, on configure des zones de recherche ayant des tailles variables et adaptées au modèle de mobilité et au taux d'appels entrants qui sont différents pour chaque mobile, comme illustré dans Figure 19.6.

En générale chaque mobile suit un modèle de mobilité qui est différent des autres. Chaque utilisateur mobile se déplace à une vitesse qui change en fonction du temps. De plus chaque utilisateur choisit des directions différentes des autres. Recherche de mobiles adaptative est basée sur cette observation. On configure des zones de recherche différentes pour chaque mobile, adaptées à son modèle de mobilité et taux d'appels entrants. Ceci consiste à optimiser :

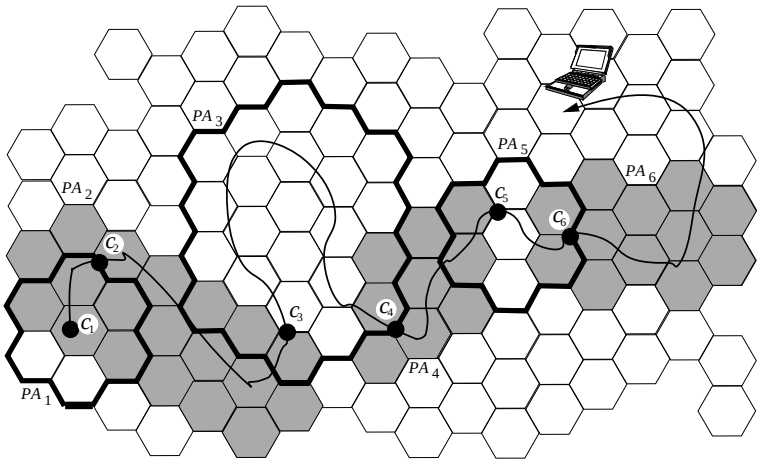


Figure 19.6: Recherche de mobile adaptative. Les formes des zones sont adaptées au modele de mobilité.

- la taille des zones
- la forme des zones

Un algorithme pour l'optimisation de la taille des zones de recherche a déjà été proposé dans la littérature. L'algorithme consiste à augmenter la taille d'un mobile lorsqu'il se déplace à une grande vitesse. Ceci permet de réduire le taux des messages de mise à jour de localisation, et la consommation de l'énergie. Lorsqu'un mobile réduit sa vitesse, sa zone de recherche est également réduite, ce qui permet au système de réduire le coût de la recherche.

19.4.2 Problématique et notre approche

Dans cette partie de la thèse on s'est intéressé à l'optimisation de la forme des zones de recherche. Ceci nécessite un algorithme de configuration automatique. Pour commencer on a comparé deux possibilités :

- zones configurées par les mobiles
- zones configurées par le réseau

Dans le premier cas, chaque mobile apprend les sous-réseaux qu'il visite et garde un historique de mobilité. L'historique de mobilité fournit les informations nécessaires pour la configuration des zones. Cependant, cette approche ne passe pas à l'échelle. La convergence des zones dépend du nombre d'échantillons (déplacements) que l'on a dans l'historique de mobilité. Par conséquent, lorsqu'un mobile visite un endroit pour la première fois, il n'aura pas de service de recherche à sa disposition puisqu'il ne connaît pas les identités de sous-réseaux dans la région nouvellement visitée.

Dans cette thèse on a adopté la deuxième approche qui est développée dans la section suivante.

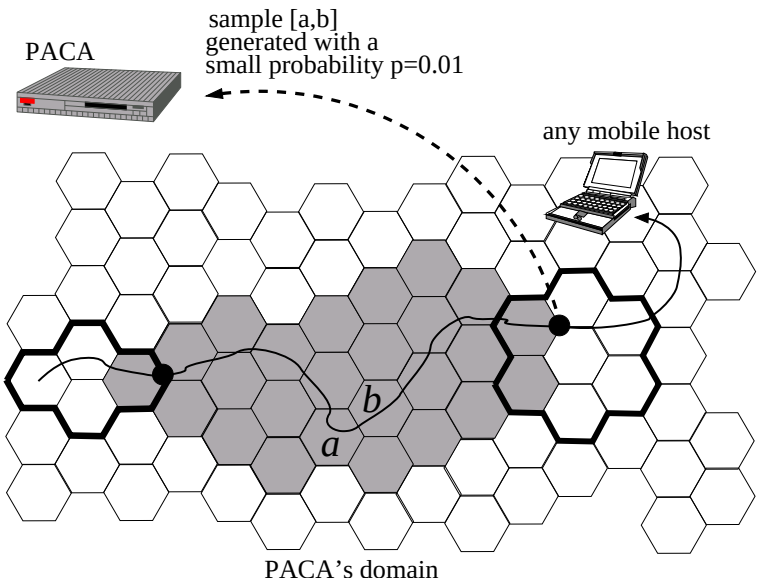


Figure 19.7: Algorithme pour la configuration automatique des zones de recherche. (PACA: abbréviation en anglais pour "agent de configuration des zones de recherche").

19.4.3 Configuration automatique des zones adaptatives

L'algorithme de configuration qu'on a conçu nécessite l'introduction d'une nouvelle entité que nous appelons un *agent de configuration de zones de recherche*. Le réseau est organisé en plusieurs domaines et chaque agent de configuration est responsable de la configuration des zones de recherche dans son domaine. Lorsqu'un mobile se déplace d'un sous-réseau A à un autre sous- réseau B, il envoi à son agent de configuration, avec une très petite probabilité, l'adresse de ces deux sous-réseaux (A,B). Tous les mobiles participent à ce processus d'échantillonnage. Ces échantillons permettent à l'agent de configuration de calculer la probabilité des directions dans chaque sous-réseau. A partir de cette information il est possible de calculer des zones de recherche de formes optimales.

Cet algorithme est illustré dans Figure 19.7.

19.5 Recherche de mobiles de bout en bout

Dans cette partie de la thèse on propose un protocole de recherche de bout en bout pour un support de mobilité plus robuste et respectant la vie privée des utilisateurs. Notre approche permet à un correspondent, avec une forte probabilité, de localiser un mobile même si son agent mère n'est pas disponible.

Cette approche est basée sur l'observation suivante : en générale un utilisateur mobile a une *zone de vie* à peu près définie. Par exemple, une zone de vie peut couvrir les endroits préférés de l'utilisateur, le lieu travail, domicile, etc. Etant donné que le hôte mobile à une zone de vie à peu près fixe, le mobile devrait pouvoir apprendre l'ensemble des sous-réseaux qui couvrent sa zone de vie.

Recherche de mobile de bout en bout a deux motivations importantes:

- Robustesse à la défaillance de l'agent mère:

  Dans la gestion de la mobilité IPv6, on a un réseau mère pour chaque mobile. Chaque mobile dispose d'une adresse fixe dans son réseau mère, connue par tous ses correspondants potentiels. Lorsqu'un hôte mobile se déplace vers un nouveau réseau IP, il configure une deuxième adresse IP ou le hôte est temporairement joignable. Afin de recevoir des paquets, le hôte mobile doit tenir au courant un routeur nommé agent mère dans son réseau mère, de son adresse temporaire courante. Ce mécanisme nécessite la transmission d'un paquet de contrôle que l'on appelle mise à jour de localisation. Lorsqu'un correspondant ouvre une session avec un destinataire mobile, il envoie son paquet à l'adresse mère du destinataire qui est publiquement connue. Alors, l'agent mère du mobile intercepte le paquet et le route vers l'adresse temporaire courant du mobile. Le mobile reçoit le paquet de son correspondant, et la communication de bout en bout commence.

  En réalité, on ne peut pas toujours faire l'hypothèse d'un agent mère 100% robuste. En cas de panne du réseau mère ou de l'agent mère, un mobile deviendra injoignable à tous ses correspondants. En théorie, le correspondant devrait être capable de localiser le mobile lui-même puisqu'il existe un chemin plus court entre le correspondant et le mobile.

- Confidentialité de la localisation:

  Dans la gestion de la mobilité IPv6, un mobile tient au courant son agent mère de tous ces mouvements. Dans certain cas, cela peut être indésirable. En remplaçant (temporairement) la fonction de l'agent mère par la recherche de bout en bout un mobile peut arriver à préserver la confidentialité de ses mouvements.

  Cependant, il est important de noter que la recherche de bout en bout n'est possible que dans la zone de vie du mobile, et peut être effectué que par ses correspondants critiques ayant obtenu cette information du mobile.

19.5.1 Configuration des zones de vie

Contrairement à notre hypothèse dans Section 19.4.2, ici on peut raisonnablement supposer que les mouvements dans une zone de vie sont répétitifs et que chaque mobile peut apprendre sa propre zone de vie.

L'algorithme qu'on a développé consiste à garder un historique de mobilité à partir duquel on obtient les probabilités des directions possibles dans chaque sous-reséau de la zone de vie. A partir de ces probabilités, on arrive à obtenir un ensemble de sous-réseaux de forme optimale. La construction de la zone de vie consiste à ajouter récursivement à la zone de vie, le sous-réseau voisin qui serait visité avec la plus grande probabilité, en quittant la zone de vie. Un exemple est illustré dans Figure 19.8.

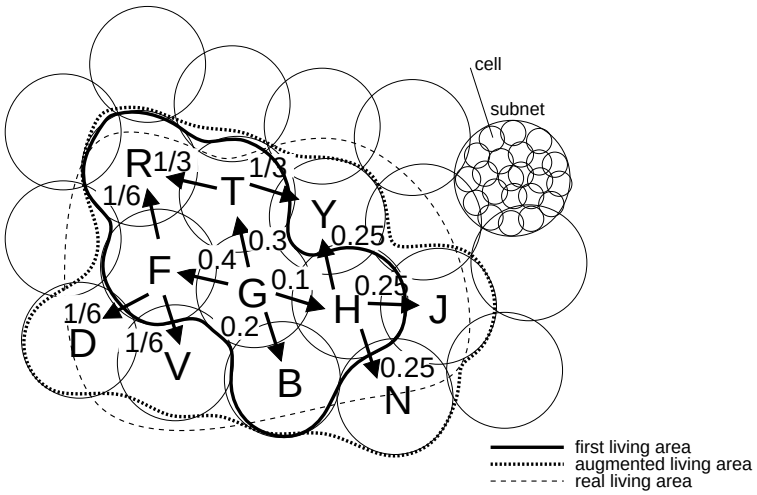


Figure 19.8: Algorithme pour la configuration d'une zone de vie.

19.5.2 La définition du protocole

Ayant configuré sa zone de vie, un mobile la transmet à ses correspondants critiques. Une zone de vie est représentée par les 64-bit identifiants des sous- réseaux qui la constituent. On fait l'hypothèse que chaque mobile à un identifiant d'interface fixe dans sa zone de vie. Soit A, B, C et D les 64-bit identifiants de la zone de vie d'un mobile, et x son identifiant d'interface (64-bit). Le mobile transmet cette information à tous ces correspondants critiques.

A partir de cette information, un correspondant peut rejoindre le mobile en transmettant ses paquets aux adresses suivantes:

$$A|x$$

$$B|x$$

$$C|x$$

$$D|x$$

comme illustré dans Figure 19.9. Le protocole qu'on a décrit ci-dessus ne nécessite pas un support particulier de la part du réseau. Cependant, une petite modification aux routeurs d'accès peut facilement améliorer la vitesse de la recherche de bout en bout. On propose qu'un routeur d'accès dans la zone de recherche, à la réception d'un paquet en provenance d'un correspondant, essaie de contacter le mobile dans son sous- réseau. Si le mobile ne répond pas tout de suite, le routeur d'accès retourne au correspondant un message indiquant *destination probablement injoignable*, mais continue quand même a chercher le mobile dans son sous-réseau. A la réception d'un paquet de ce type, le correspondant peut envoyer son paquet à une différente candidate adresse du mobile destinataire. Cette optimisation est illustré dans Figure 19.10.

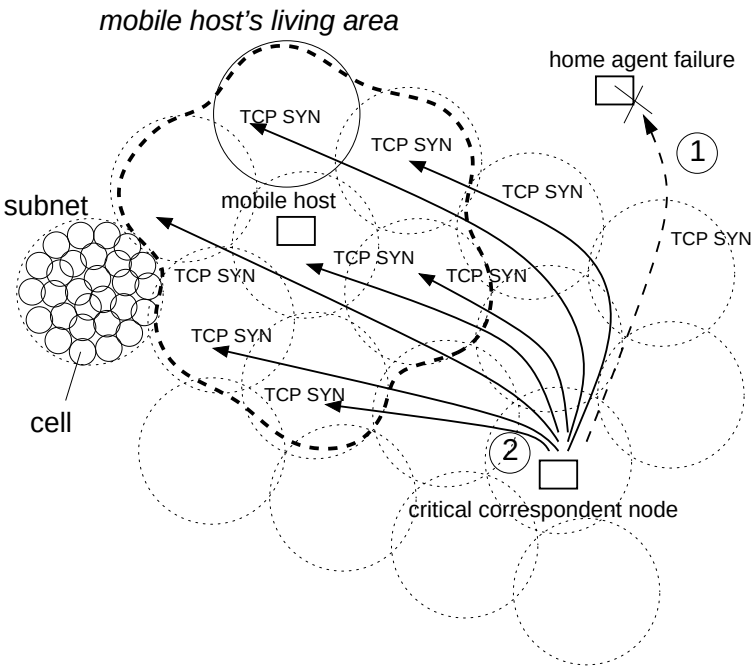


Figure 19.9: Ouverture d'une session TCP en cas de défaillance, en utilisant l'algorithme de recherche de bout en bout.

19.6 Conclusion

Dans cette thèse, on a développé des algorithmes et protocoles afin de réduire la consommation de la bande passante et de l'énergie dans un Internet mobile.

Dans un Internet future accueillant des milliards de hôtes hautement mobiles, des paquets de mise à jour de localisation peuvent consommer une partie importante de la bande passante dans l'Internet. Dans la littérature deux différentes approches ont été proposées afin de résoudre ce problème. Dans la première approche on configure des sous-reséaux couvrant un grand nombre de cellules, et donc une zone géographique large. Ainsi, la fréquence des mouvements IP des mobiles sont réduites. La deuxième solution consiste à regrouper plusieurs sous-réseaux pour former des zones géographiques nommées des zones de recherche IP. Un hôte mobile qui n'est pas en cours de communication appelé hôte dormant, tient au courant son agent mère de sa zone de recherche courante, au lieu de sa localisation exacte.

Le problème commun dans ces deux approches c'est le coût que l'on doit payer pour la diffusion fréquente des messages tels que la sollicitation de voisin ou recherche de mobile. On a proposé un nouvel algorithme de diffusion basée sur les filtres de Bloom. La même procédure à été appliquée à deux problèmes différentes, c'est-à-dire à la découverte de voisin et à la recherche de mobile.

Nous avons également développé une architecture et un algorithme distribué pour la configuration automatique des zones de recherche. Cette architecture permet également d'implémenter des algorithmes de recherche adaptative afin de mieux adapter la taille et la forme des zones de recherche au modèle de mobilité propre à chaque mobile.

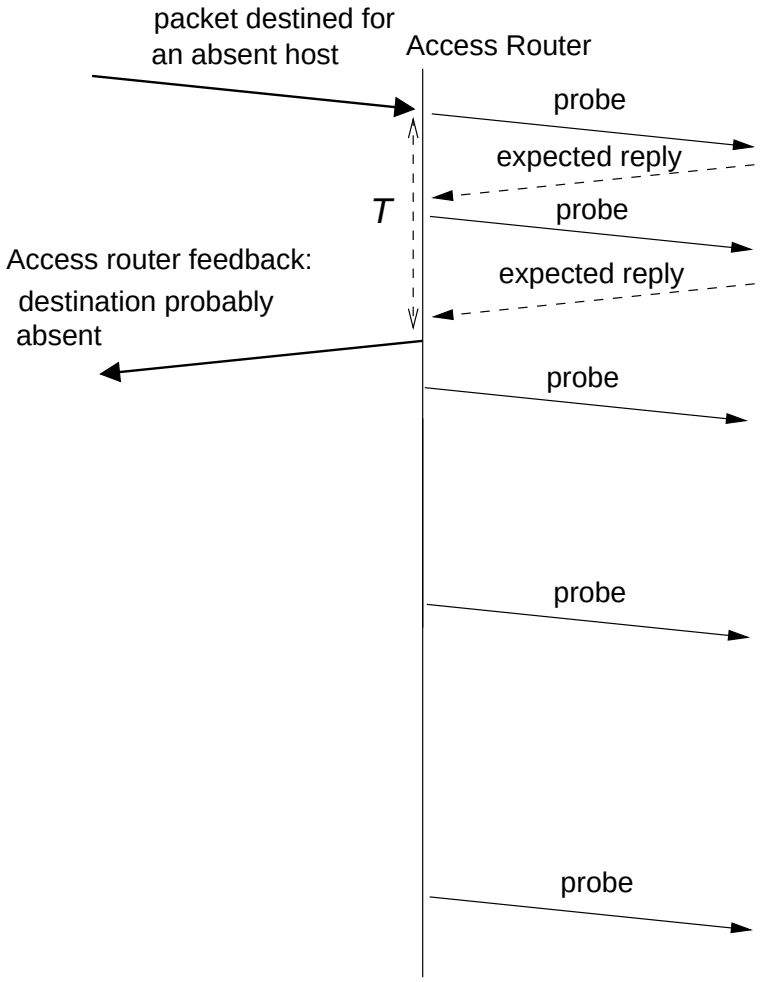


Figure 19.10: Support du réseau pour la réduction du temps de recherhe.

Finalement, on a développé un protocole de recherche de mobile de bout en bout. Ce protocole permet à un correspondant de localiser un mobile, si par exemple ce dernier est devenu injoignable à cause d'une défaillance d'agent mère. Le protocole n'est applicable que dans une zone géographique limitée que l'on appelle une zone de vie. La plupart des utilisateurs mobiles ont leur propre zone de vie qu'ils ne quittent que rarement. Cette observation implique que l'on peut en général parvenir à localiser un mobile sans le support du réseau.

Bibliography

[1] J. Postel, "Internet Protocol," RFC 791, IETF, September 1981.

[2] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2440, IETF, December 1998.

[3] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," RFC 2373, IETF, July 1998.

[4] "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority," http://standards.ieee.org/db/oui/tutorials/eui64.html, IEEE, 1997.

[5] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration," RFC 2462, IETF, December 1998.

[6] R. Droms et al., "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," RFC 3315, IETF, July 2003.

[7] A. Conta and S. Deering, "Generic Packet Tunneling in IPv6 Specification," RFC 2473, IETF, December 1998.

[8] T. Narten, E. Nordmark, and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)," RFC 2461, IETF, December 1998.

[9] A. Conta and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," RFC 2463, IETF, December 1998.

[10] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, July 1970.

[11] J. K. Mullin, "Optimal semijoins for distributed database systems," *IEEE/ACM Transactions on Software Engineering*, vol. 5, no. 16, May 1990.

[12] 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*, October 1998, vol. 28, pp. 254–265, Corrected version available at URL: http://www.cs.wisc.edu/~cao/papers/summarycache.html.

[13] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters," http://citeseer.nj.nec.com/543187.html.

[14] W. C. Feng et al., "Stochastic fair blue: A queue management algorithm for enforcing fairness," in *Proceedings of INFOCOM'01 Conference*, Los Alamitos, CA, April 2001, pp. 1520–1529.

[15] B. Gronvall, "Scalable multicast forwarding," http://www.acm.org/sigcomm/ccr/archive/2002/jan02/ccr-sc01-posters/bjorngronvall.ps.

[16] C. Huitema, *IPv6: The New Internet Protocol*, O'Reilly Networking, 1996.

[17] B. O'Hara et al., *The IEEE 802.11 Handbook: A Designer's Companion*, Prentice Hall, 1996.

[18] K.P. Pearson, "Fast hashing of variable-length test strings," *CACM*, vol. 33, no. 6, pp. 677–680, June 1990.

[19] R. Gilligan et al, "Basic Socket Interface Extensions for IPv6," RFC 2553, IETF, March 1999.

[20] "How to Protect Your Network Against the Nimda Virus," http://www.cisco.com/warp/public/63/nimda.shtml.

[21] "Harmony 7560 AP Controller: How to Set the ARP and Multicast Filters ," http://www.proxim.com/support/all/harmony/technotes/tn2001-08-10a.html.

[22] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6," Internet Draft, work in progress, IETF, October 2002.

[23] P. Nikander, J. Kempf, and E. Nordmark, "IPv6 Neighbor Discovery trust models and threats," Internet Draft, work in progress, IETF, April 2003.

[24] R. Koodli et al., "Fast Handovers for Mobile IPv6," Internet Draft, Work in Progress, IETF, September 2003.

[25] N. Moore, "Optimistic Duplicate Address Detection," Internet Draft, Work in Progress, IETF, August 2003.

[26] I. Soto et al., "Random generation of interface identifiers," Internet Draft, Work in Progress, July 2002.

[27] A. Snoeren and H. Balakrishnan, "An End-to-End Approach to Host Mobility," in *6th International Conference on Mobile Computing and Networking (MobiCom)*, Boston, Massachusetts, August 2000.

[28] M. Mouly and M.B. Pautet, *The GSM System for Mobile Communication*, ISBN: 2-9507190-0-7, 1992.

[29] Digital cellular communication system, "General Packet Radio Service, Service Description - Stage 2," Tech. Rep., GSM 03.60, Version 6.0, ETSI, 1998.

[30] IS-657, "Packet Data Service Option for Wideband Spread Spectrum Systems," Tech. Rep., 1996.

[31] 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.

[32] R. Ramjee et al, "HAWAII: A Domain-based Approach for Supporting Mobility in Wide-area Wireless Networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, June 2002.

[33] R. Ramjee et al, "IP Paging Service for Mobile Hosts," in *Proceedings of MOBICOM'2001*, Rome, Italy, July 2001.

[34] C. Perkins, "IP Mobility Support," RFC 3344, IETF, August 2002.

[35] X. Zhang, J. Castellanos, and A. Campbell, "Design and Performance of Mobile IP Paging," *ACM Mobile Networks and Applications (MONET)*, vol. 7, no. 2, March 2002.

[36] J. Kempf, "Dormant Mode Host Alerting (IP Paging) Problem Statement," RFC 3132, IETF, June 2001.

[37] J. Kempf et al, "Requirements and Functional Architecture for an IP Host Alerting Protocol," RFC 3154, IETF, August 2001.

[38] J. Kempf and P. Mutaf, "Mobile IPv6 and IP Paging for Dormant Mode Location Update," in *IEEE Wireless Communications and Networking Conference (WCNC'03)*, New Orleans, Lousiana, March 2003.

[39] C. Castelluccia, "Extending Mobile IP with Adaptive Individual Paging: A Performance Analysis," *ACM Mobile Computing and Communication Review (MC2R)*, April 2001.

[40] P. Mutaf and C. Castelluccia, "Hash-based Paging and Location Update Using Bloom Filters," *to appear in ACM Mobile Networks and Applications (MONET)*.

[41] P. Mutaf and C. Castelluccia, "A Hash-Based Paging and Location Update Procedure," in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Sophia-Antipolis, France, March 2003.

[42] P. Mutaf and C. Castelluccia, "DPAC: Dynamic Paging Area Configuration," draft-mutaf-dpac-00.txt, Internet Draft, Work in Progress, September 2001.

[43] C. Castelluccia and P. Mutaf, "An Adaptive Per-host IP Paging Architecture," *ACM SIGCOMM Computer Communication Review (CCR) Special Issue on Wireless Extensions to the Internet*, October 2001.

[44] P. Nikander et al., "Mobile IP version 6 (MIPv6) Route Optimization Security Design ," in *IEEE Semiannual Vehicular Technology Conference*, Orlando, Florida, October 2003.

[45] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," RFC 2401, IETF, November 1998.

[46] R. Atkinson, "IP Authentication Header," RFC 1826, IETF, August 1995.

[47] R. Atkinson, "IP Encapsulating Security Payload (ESP)," RFC 1827, IETF, August 1995.

[48] C. Kaufman, "Internet Key Exchange Protocol (IKEv2)," Internet Draft, work in progress, IETF, April 2003.

[49] P. Nikander, "An address ownership problem in ipv6," Internet Draft, Work in Progress, February 2001.

[50] G. Montenegro and Castelluccia C., "Statistically Unique and Cryptogrpahically Verifiable Identifiers and Addresses," in *Network and Distributed System Security*, San Diago, CA, February 2002.

[51] G. O'Shea and M. Roe, "Child-proof authentication for mipv6 (cam)," *ACM Computer Communications Review*, April 2001.

[52] P. Mutaf and C. Castelluccia, "IP Paging Security Requirements," Internet Draft, work in progress, IETF, http://www.inrialpes.fr/planete/people/mutaf/Publications.html, May 2001.

[53] P. Mutaf and Castelluccia C., "Insecurity of the Paging Channel in a Wireless Internet," in *IEEE Applications and Services in Wireless Networks*, Bern, Switzerland, March 2003.

[54] V. Paxson, "An analysis of using reflectors for distributed denial-of-service attacks.," *computer Communication Review*, vol. 31, no. 3, July 2001.

[55] J. Rosenberg et al, "SIP: Session Initiation Protocol," RFC 3261, IETF, June 2002.

[56] J. Postel, "User Datagram Protocol," RFC 768, IETF, August 1980.

[57] J. Postel, "Transmission Control Protocol," RFC 793, IETF, September 1981.

[58] Morris, R., "A Weakness in the 4.2BSD Unix TCP/IP Software," Bell Labs Computer Science Technical Report 117, February 1985.

[59] Computer Emergence and Response Team (CERT), "CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks," http://www.cert.org/advisories/ca-1996-21.html, November 2000.

[60] "SYN-cookies," http://cr.yp.to/syncookies.html, 2003.

[61] Cowzilla, and Pixel Dreamer, "The Puke attack," http://www.rs-labs.com/papers/tacticas/ircutils/puke.html.

[62] T. Aura et al., "DOS-resistant Authentication with Client Puzzles," in *Security Protocols, 8th International Workshop*, Cambridge, UK, April 3-5, 2000.

[63] J. Postel and J. Reynolds, "File Transfer Protocol," RFC 959, IETF, October 1985.

[64] J. Postel, "Simple Mail Transfer Protocol," RFC 821, IETF, August 1982.

[65] R. Fielding et al, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, IETF, June 1999.

[66] J. Myers and M. Rose, "Post Office Protocol - Version 3," RFC 1939, IETF, May 1996.

[67] M. Crispin, "Internet Message Access Protocol," RFC 2060, IETF, December 1996.

[68] S. Bellovin et al., "ICMP Traceback Messages," Internet Draft, work in progress, IETF, February 2003.

[69] Snoeren et al., A.C., "Hash-Based IP Traceback," Bbn technical memorandum no. 1284, http://www.ir.bbn.com/documents/techmemos/tm1284.ps.

[70] Savage et al., S., "Practical Network Support for IP Traceback," Technical report uw-cse-2000-02-01, department of computer science and engineering, university of washington, http://www.cs.washington.edu/homes/savage/traceback.html.

[71] W. R. Cheswick and S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker, first edition*, Addison-Wesley, 1994.

[72] A. Krywaniuk, "Security Properties of the IPsec Protocol Suite," Internet Draft, work in progress, IETF, draft-ietf-ipsec-properties-02.txt, July 2002.

[73] S. Bellovin, "Guidelines for Mandating the Use of IPsec," Internet Draft, work in progress, IETF, October 2002.

[74] C. Perkins et al, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561, IETF, July 2003.

PUBLICATIONS

Journal, Conference and Workshop publications

- **Hash-Based Paging and Location Update Using Bloom Filters** ***(A paging algorithm that is best suitable for IPv6)***.
  Pars Mutaf and C. Castelluccia.
  To appear in ACM/Kluwer Journal on Mobile Networks and Applications (MONET).
  2003.

- **Insecurity of the Paging Channel in a Wireless Internet**.
  Pars Mutaf and C. Castelluccia.
  IEEE Workshop on Applications and Services in Wireless Networks (ASWN'03), Berne, Switzerland.
  March 2003.

- **A Hash-Based Paging and Location Update Procedure**.
  Pars Mutaf and C. Castelluccia.
  Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt'03), Sophia-Antipolis, France.
  March 2003.

- **Mobile IPv6 and IP Paging for Dormant Mode Location Update**.
  James Kempf and Pars Mutaf
  IEEE Wireless Communications and Networking Conference, New Orleans, Lousiana, USA.
  March 2003.

- **An Adaptive Per-Host IP Paging Architecture**.
  Claude Castelluccia and Pars Mutaf
  ACM SIGCOMM Computer Communication Review (CCR).
  Special Issue on Wireless Extensions to the Internet.
  October 2001.

- **Defending against a Denial-of-Service Attack on TCP**.
  Pars Mutaf.
  Second International Symposium on Recent Advances in Intrusion Detection (RAID'99), CERIAS/Purdue, Indiana, USA.
  September 1999. *Presentation sponsored by IBM Business Recovery Services/Emergency Response Service and the SANS Institute (RAID'99 Corporate Sponsors).*

IETF RFCs and Internet Drafts

- **Requirements and Functional Architecture for an IP Host Alerting Protocol**.
  J. Kempf, C. Castelluccia, P. Mutaf, N. Nakajima, Y. Ohba, R. Ramjee, Y. Saifullah, B. Sarikaya, X. Xu.

RFC 3154, IETF (Internet Engineering Task Force)
August 2001.

- **IP Paging Security Requirements**.
  P. Mutaf, C. Castelluccia.
  IETF Internet Draft Submitted to the Seamoby Working Group (Our contribution to RFC 3154).
  May 2001.

- **DPAC: Dynamic Paging Area Configuration**.
  P. Mutaf, C. Castelluccia.
  IETF Internet Draft Submitted to the Seamoby Working Group.
  August 2001.

TALKS

- **DPAC: Dynamic Paging Area Configuration**.
  IP-based Cellular Networks (IPCN'02), Paris, France.
  April 2002.

- **IP paging security problems**.
  IP-based Cellular Networks (IPCN'01), Paris, France.
  April 2001.