UNIVERSITÉ DE LIÈGE
Faculté des Sciences Appliquées
Département d'Électricité,
Électronique et Informatique

Performing and Making Use of Mobility Prediction

Thèse présentée par

Jean-Marc François

en vue de l'obtention du titre de Docteur en Sciences de l'Ingénieur

Année académique 2006–2007

Abstract

Mobility prediction is defined as *guessing the next access point(s)* a mobile terminal will join so as to connect to a (wired or wireless) network. Knowing in advance where a terminal is heading for allows taking proactive measures so as to mitigate the impact of handovers and, hence, improve the network QoS. This thesis analyzes this topic from different points of view. It is divided into three parts.

The first part evaluates the *feasibility of mobility prediction* in a real environment. It thus analyzes a mobility trace captured from a real network to measure the intrinsic entropy of the nodes motion and to measure the effectiveness of a simple prediction method.

The second part investigates *how to perform mobility prediction.* Firstly, it examines a generic prediction scheme based on a simple machine learning method; this scheme is evaluated under various conditions. Secondly, it shows how the pieces of information that are most useful for the prediction algorithm can be obtained.

The third part studies how knowing the probable next access point of a mobile terminal allows one to *improve the QoS* of the network considered. We deal with two situations. We first show how the handover blocking rate of a cellular network can be decreased thanks to resource reservation. We then propose a new routing protocol for delay tolerant networks (*i.e.* an ad hoc network where packets must be delayed in the absence of an end-to-end path) that assumes that the contacts between the nodes can be (imperfectly) predicted.

Acknowledgments
Remerciements

My sincere acknowledgments towards the members of my thesis committee, namely Chris Blondia, Serge Fdida, Ignas Niemegeers, Louis Wehenkel and Pierre Wolper, for their attentive review of this thesis and their interest in my work.

(in French)

Mes vifs remerciements vont à M. Guy Leduc, Professeur à l'Université de Liège et promoteur de cette thèse. Il a su, par son inépuisable énergie et ses conseils avisés, être le coach efficace qui m'a guidé durant ces cinq années. Qu'il soit remercié de sa disponibilité, de son ouverture d'esprit et de ses nombreux encouragements.

Merci aux membres de l'unité de recherche en réseaux informatiques: Charline, Cyril, Fabian, François, Gaël, Geoffroy, Ibtissam, Jean, Laurent, Nicolas, Olivier, Sandrine, Simon et Sylvain. Ils m'ont souvent fourni leurs conseils avisés et permis de travailler dans une ambiance de travail agréable et stimulante.

Plus globalement, merci à toutes les personnes que mon passage à l'Université m'a permis de rencontrer. Grâce à eux ces dernières années furent une expérience hors du commun.

Cette thèse n'aurait sans doute jamais vu le jour sans le support quotidien de ma famille et de mes proches. Sachez l'importance des encouragements que vous m'avez donnés et de l'attention que vous m'avez portée.

Contents

List of Figures

1 Introduction

In the beginning the Universe was created. This has made a lot of people very angry and been widely regarded as a bad move.

—DOUGLAS ADAMS,
The Restaurant at The End of the Universe (1980)

1.1 Context

1.1.1 Overview

In 1895, following the work of Heinrich Hertz, Guglielmo Marconi demonstrated a radio communication for the first time. He surely could not forecast how popular his invention would become: he thought that Morse codes were adequate for communication between ships and that voice communications were not commercially viable. He left the early experimentation with wireless telephony to Reginald Fessenden and Lee de Forest; they succeeded to transmit speech over a distance of 1.5km on December 1900 ([Bel95]).

Antenna-to-antenna communications are now commonplace in our everyday lives.

In 1985, the number of mobile phone users in the US was estimated to 203,000. The million of users milestone was reached in 1988, 10 million in 1993. There were 100 million users in 2001, and about 220 million this year, 2006. The number of subscribers this year was up more than 24.9 million, an increase of 14% compared to 2005 ([CTI06]).

Mobile phones are surely not the only wireless device one regularly carries around. PDAs (Personal Digital Assistants) are more and more popular and usually have at least one wireless interface. Satellite phones are sold for several years.

Laptop computers are not uncommon and are virtually always able to be part of a mobile network.

Wireless devices are thus common and appear to become even more widespread in the years to come. Network connectivity is so ubiquitous — yet heterogeneous — that the concept of “ambient network” has been defined; it refers to the seamless cooperation of all the network-aware devices around us ([NSA$^+$04]).

The increased demand for connectivity now meets a high number of wireless standards: ZigBee (for embedded, typically cheap, devices such as sensors), BlueTooth (typically to replace cables connecting various devices), irDA (offers infrared connections), 2G/3G mobile phones, IEEE 802.11 (or WiFi standard, for Wireless LANs — WLANs). Some of those standards propose different physical layers; for example, IEEE 802.11 networks currently exist in 3 flavours: 802.11a, 802.11b and 802.11g.

The standardization process is not likely to terminate soon; new drafts are being studied and will be available in a few year: IEEE 802.16 (or WiMax, which proposes to replace DSL connections), IEEE 802.15 (high speed — about 1Gbps — short range Wireless Personal Area Network or WPAN), IEEE 802.11n (a flavour of WiFi using multiple antennas and which promises to deliver 100Mbps).

The number of QoS-dependent applications on packet switched networks is rising fast; *QoS* stands for *Quality of Service*, meaning that delay, delay jitter and packet losses can be controlled. The predominance of Internet has transformed it in a general-purpose information transmitting medium that can be used not only to transfer files and emails, but also films, television, radio, songs, phone calls and video conferences. Revenue generated by Voice over IP (VoIP) in the US, Europe and Asia Pacific have doubled in 2005 and are forecast to continue booming during several years ([Inf06]). Those applications are bandwidth-demanding and, furthermore, interactive sessions (*e.g.* phone and video conferences) do not tolerate round-trip delays greater than about 100ms ([Che96]). IP-level solutions have been developed by the IETF to guarantee quality of service; the most well-known solution are DiffServ ([BBC$^+$98]) and IntServ ([BCS94] and versions adapted to mobile networks [TBA01, TLLW03]).

The convergence of bandwidth-hungry services towards IP networks forces wireless network providers to find ways to increase the capacity of their networks. In cellular networks, an often used method is to reduce the antenna coverage so as to increase the spatial frequency re-use (*i.e.* signals between antennas are modulated in a certain frequency range; those frequencies are used by several antennas at the same time. Thanks to the signal fading, they don’t interfere if they are far apart; see [Tab00]).

In wireless networks, service degradation has multiple sources.A first problem comes from the very nature of electromagnetic modulated waves. This media has characteristics that can vary quickly with time because of interferences (*e.g.* because of frequency re-use, see above), path loss (if we denote d the distance between antennas, the received power density varies in $1/d^2$), shadowing effects (*i.e.* the effect of obstacles and clutter), Rayleigh fading (the effect of destructive interference of the signal propagation on different paths)...

Access points (APs, often called *Base Stations*, BSs, in cellular networks) are the layer 2[1] devices offering wireless connection with the mobile nodes. A *handover* (or, equivalently, *handoff*), is the transfer of a communication from one AP to another AP as the mobile moves. Since the traffic has to be re-routed, handovers are another important cause of service degradation. As we have seen, the coverage area of antennas tends to reduce, and, thus, handovers are likely to become more and more frequent in the future[2].

Handovers can be *hard* or *soft*, *horizontal* or *vertical*. During a *hard* handover, the mobile never communicates with both the old and the new antennas (*e.g.* GSM networks perform hard handovers); during a *soft* handover, the mobile maintains its connectivity with two antennas (this is particularly suited to certain modulation techniques such as W-CDMA). A handover between access points of different technologies (*e.g.* GSM to WLAN) is said *vertical*; otherwise it is *horizontal*. A complete definition of mobility-related terms can be found in [MK04].

The exact impact of handovers depends largely on the kind of network studied, but, in every kind of packet-switched networks, handovers can cause a serious quality of service degradation ([CZD$^+$04]). We will not, in this introduction, enumerate all the possible handover mechanisms, nor study how they alter QoS; the reader interested in an overview of layer 2 handover mechanisms can refer to [TRV98, BR02]. We here prefer to scrutinize the representative example of WiFi networks.

1.1.2 Handovers at layer 2

WiFi networks are composed of several addressable entities. The access points connect the wireless network to a fixed infrastructure (often called *distribution system*, DS, in WiFi parlance). The *mobile stations* (STAs, also called *mobile hosts*, MHs, or *mobile nodes*, MNs, or *mobile terminals*, MTs) move and, therefore, can roam from one AP to another ([Hay99]).

WiFi networks operate in the unlicensed 2.4GHz (802.11b and g) and 5GHz (802.11a) radio bands. Each band is divided into channels (for example, in Europe, the 2.4GHz band is composed of 13 overlapping 22MHz channels). When a terminal communicates with an AP, they use a given channel; they interfere with nearby channels[3].

[1] Layer 2 or *link layer*. Network protocols are divided into layers defined by the ISO; see [Zim80] for more details.

[2] New modulation techniques such as WCDMA and cdma2000 also tend to increase the handover rate ([KMR04]).

[3] Distant channels are supposed independent; in practice, the poor selectivity of WiFi interfaces might weaken this hypothesis ([MSA03]).

Many WiFi manufacturers use the *relative signal strength with hysteresis and threshold* (RSSHT, see [GPZ05]) handover scheme. When a mobile moves away from the AP it is associated with, the measured Signal to Noise Ratio (SNR), $\mathrm{SNR_{old}}$, decreases until it reaches a predefined threshold. The mobile then starts scanning for new APs characterized by an SNR value $\mathrm{SNR_{new}}$ such that

$$\mathrm{SNR_{new}} - \mathrm{SNR_{old}} > \Delta\mathrm{SNR}$$

where ΔSNR is the hysteresis SNR.

Scanning all channels to find a new AP takes a lot of time and is, in fact, the main contributing factor to the handover delay. The total handover time thus largely depends on the way this scanning is implemented, and thus depends on the manufacturers of both the access point and the interface of the mobile node. Other factors impact the handover latency, such as, for example, the distance between the APs ([MSA03]).

Figure 1.1 shows the result of an experiment meant to illustrate the delay introduced by handovers. A fixed source sends UDP packets to a mobile destination by means of a WiFi network. Packets are numbered and sent every 30ms. Two APs are connected to the packet source *via* an Ethernet switch. The APs' antennas are connected to electronically-driven attenuators. The MH motion is simulated by means of those attenuators.

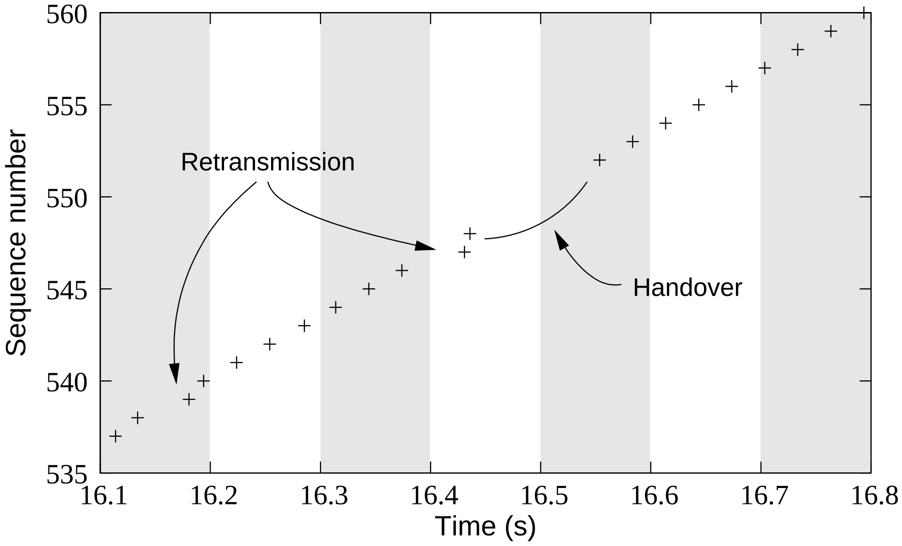


Figure 1.1: Impact of WiFi handovers. Some frames experience transmission errors and are re-transmitted by the AP until acknowledged by the receiver (or dropped); this causes delay jitter. Retransmissions are initiated by the access point. This experiment has been realized with standard, off-the-shelf devices (Cisco 2500 APs, Orinoco 802.11b PCMCIA card).

Before time 16.45s, the mobile is connected to the first AP. Its signal has been attenuated, causing a low SNR and transmission errors. Three frames have been lost during the handover, causing a delay of about 100ms. Considering a phone conversation, this is admissible but already considerable since no IP-level re-routing is required in this simple scenario.

1.1.3 Handovers at higher levels

This chapter focuses on general mobility protocols for IP networks. Several have been proposed (*e.g.* Sony, IBM, Columbia Mobile Host Protocol; see [MS93]), but the most well-known is undoubtedly Mobile IP ([Per97, JP03]). The reader interested in other layer 3 mobility-support paradigms can refer to [FHL06], a recent review authored by Le et al.

The success of Mobile IP can certainly be attributed to its simplicity and scalability. It is formally described in RFC 2002 (see [Per96b]; other related protocols are defined in [Per96a, Per96c, Sol96, CHP96]). It is briefly described below; we do not distinguish Mobile IPv4 and Mobile IPv6.

Each mobile has a *home network* and can connect itself to *Foreign Networks* (FN). A *Home Agent* (HA) is connected to the home network and a *Foreign Agent* (FA) resides on the foreign network. Once the mobile is connected to a foreign network, it is informed of the presence of a mobility agent (MA); it can also actively search that agent. Notice that the mobile motion does not necessarily imply that it uses a wireless network; it could be, for example, a laptop that is unplugged, moved, and plugged somewhere else.

Once the MA has been found, the MH uses it as a FA and obtains an address on the foreign network; this address is named *Care-of Address* (CoA). This address is used to *locate* the MH; its original (home) address is used to *identify* it.

Once its CoA is known, the MH can send it to its home agent; after that registration, the HA is able to locate the MH. Security measures must be taken so that no one can usurp the identity of the mobile.

Once this is done, if a *correspondent node* (CN) sends a packet to the MH, the HA receives this packet (using *gratuitous* and *proxy ARP* for example) and forwards it in an IP-IP tunnel ([Per96a]) to the (new) CoA. The end of this tunnel may be the FA or the MH itself (the care-of address is then said *co-located*); it decapsulates the packet and delivers it to its final destination.

Packets sent by the MH to the CN are routed normally. Since the source address of those packets belongs to the home network, they may be blocked by some firewalls.

Figure 1.2 depicts this overview of Mobile IP. Once the MH detects a new network, it can passively wait for an ICMP *Router Advertisement* (**2**) or first send a *Router Solicitation* (**1**) to force the FA to send its advertisement. Unsolicited router advertisements are sent periodically; the exact period is not determined by

Mobile IP, but should not be smaller than 3s to follow RFC 1256[4] ([Dee91]); the MH could also act as its own FA. Once a FA has been found, the MH can register its new address to the HA. This is done thanks to the *Registration Request* (**3**) and *Registration Response* (**4**) UDP messages; those packets are sent *via* the foreign agent. Once a correspondent node sends a packet to the mobile host (**5**), it is tunneled by the home agent (**6**) and decapsulated by the FA. The MN sends its response normally (**7**), creating a so-called *triangular routing*.

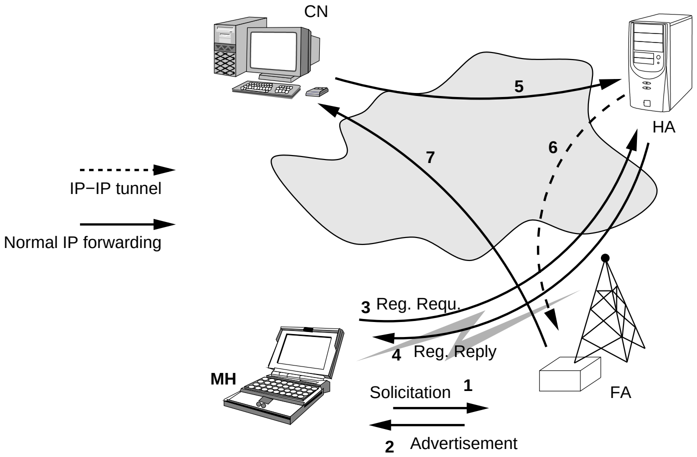


Figure 1.2: Overview of Mobile IP.

As one can see, with Mobile IP each handover implies a certain amount of signaling. Finding a foreign agent and registering with the home agent takes time: as is, Mobile IP cannot provide seamless handovers and is not suited to multimedia sessions ([FPC05]).

Some protocols aim at reducing *intra-domain* handover latency. Well-known examples of such *micro-mobility* (*i.e.* intra-domain) protocols are HAWAII ([RLT$^+$00]) and Cellular IP ([Val99, CG$^+$00]). The interested reader can refer to [RB01, RB03, CGKW02, Cas98] for an overview of those particular handover schemes. We here quickly review the popular Hierarchical Mobile IP (H-MIP) protocol ([GJP02]).

It has already been mentioned that an important part of the handover latency is caused by the registration process: each time a MH changes its point of attachment, it receives a new CoA that must be registered to its home agent. To speed up registrations, H-MIP introduces a new network entity, the Gateway Foreign Agent (GFA, sometimes called *Mobile Anchor Point* or MAP). Conceptually, this agent acts as the foreign agent of all the mobile hosts in the domain it is responsible for. The CoA of the MHs is the address of the GFA, thus as long as a MH remains in the same domain, it will perform *regional registrations* only, which is likely to greatly reduce the registration process latency (under the reasonable assumption

[4] The default value is 450s, which seems too long in this case (router advertisement as defined in RFC 1256 have not been designed specifically for Mobile IP). The Mobile IP standard also proposes to relax the constraint on the maximum number of router solicitation per second.

that most handovers are intra-domain). The GFA assigns a local care-of address to each MH, which can be used to reach the mobile as long as it is connected to the same FA.

Using H-MIP, packet routing is thus straightforward: the HA tunnels packets to the GFA. The GFA decapsulates and tunnels them again towards their final destination.

Other IP mobility protocols have been proposed. The *Session Initiation Protocol* (SIP, [RSC$^+$02]), for example, is now often used to setup voice-over-IP calls and allows mobility. It does however not solve handover-related issues: even if the precise handover latency depends on several hypotheses (it is estimated at 6s in [BBD03], or as long as 40s in [NDD03]), it is in every case way too long to allow any form of seamless motion.

1.1.4 A way towards seamless mobility

We have seen that, in the context of real-time multimedia sessions, handovers are a major hurdle on the path towards seamless mobility.

One way to tackle — or to mitigate — this problem is to guess the mobile hosts' next access point(s) so as to take pro-active actions. This guess has specific characteristics; it can take different forms and be achieved using different methods:

- As the mobile moves, it connects with several APs. Mobility prediction can forecast the next one only, or a whole sequence.

- The prediction can be done a long time in advance or just a few moments before the actual handover. It can also improve with time: uncertain at first, then more and more precise as the handover approaches.

- A MH's next AP can rarely be forecast with a 100% confidence; the probability that the MH's next AP will be the predicted one is named *correct prediction probability*. A prediction can consist of the next AP, possibly with the associated correct prediction probability. It is also possible to report a list of most probable next APs, again possibly with correct prediction probabilities.

- The prediction is inferred from some pieces of information. Those must be correlated with the handover process, and thus, most of the time, with the way the mobile moves. Those pieces of information might be generated by a specific device, such as a GPS[5].

- The prediction process can involve various protagonists: the mobile host, the access point or both.

[5] Global Positioning System: a device that allows one to track its position in real-time. Those apparatus get cheaper and cheaper and have a precision of 10m or less ([TYK05]).

The problem of forecasting a mobile host's next access point(s) is known as *mobility* (or *path*) *prediction*.

The mobiles are often small, battery-powered devices; mobility prediction mechanisms are thus normally designed to require a modest amount of processing power and wireless bandwidth.

Even if things are changing fast, low-end mobile devices (*e.g.* small GSMs) still only have a limited amount of memory. To remain general-purpose, mobility prediction schemes should thus not require to stock large quantities of data.

Once a way to perform mobility prediction has been found, it can be used to take countermeasures that allow reducing the impact of handovers (*e.g.* anticipatively reserve resources in the next AP). Going back to our example of Mobile IP (figure 1.2), one could for example take advantage of the knowledge of the MH next access point to send packets to both the current and the next APs (this process is called bi-casting, see [MS03]). This is illustrated in figure 1.3. The bicasting is initiated as soon as the HA receives (**2**) a *prediction trigger* (*i.e.* a layer/link specific event, see [MK04]). It is torn down when the MH is known to have terminated the handover.

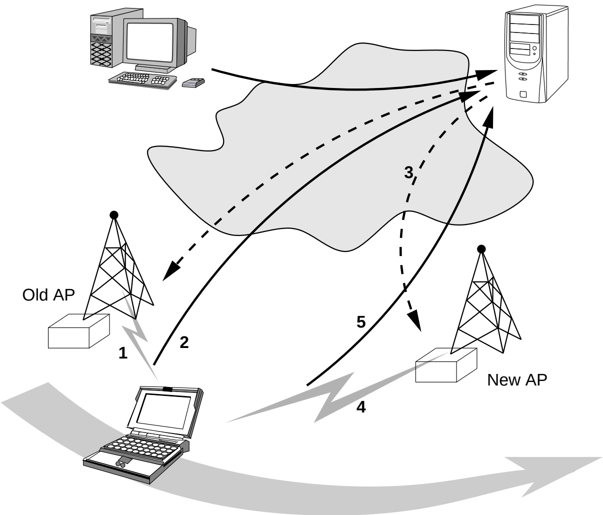


Figure 1.3: Using mobility prediction. The MH is first associated with the first AP (**1**); the home agent forwards the MH's packets in an IP-in-IP tunnel (see figure 1.2). When the prediction mechanism knows the next AP with a sufficient confidence, it sends a *prediction trigger* (**2**) to the home agent *via* the old AP. The HA the duplicates (**3**) the packets to both the old and the forecast new AP. The MH moves and eventually connects with the new AP (**4**). The MH sends the HA its updated location (**5**); the HA stops bicasting and only tears down the tunnel to the old AP.

1.2 Overview of the problems addressed by this thesis & contributions

In what follows, we give an overview of the different problems we have studied. Those problems led to original solutions, as explained below.

This overview follows the three-parts structure of this document: the first studies to what extent the motion of wireless terminals is predictable, the second how this predictability can be extracted, and the third how to take advantage of a mobility prediction mechanism.

Part I: The *Feasibility* of Mobility Prediction

We use a simple technique to model the predictability of a WiFi network. The motion of the mobiles are inferred from the mobility trace of a real, large-scale campus network. We compare the prediction ratio obtained if *(a)* the mobile host tries to guess its next AP and *(b)* if the access points predict the next AP of the mobile hosts passing by. Given that the question of who should perform the prediction is important when designing a mobility prediction scheme, it is quite surprising that such a comparison had never been undertaken before.

Part II: *Performing* Mobility Prediction

Most prediction schemes are designed for a particular type of network and make pretty strong assumptions (*e.g.* in a cellular network, a common hypothesis is the knowledge of the cells geometry). We present a generic method based on a simple artificial intelligence technique. We evaluate this method under various conditions and also tackle the particular case of wired (yet mobile) networks.

In order to guess where mobiles are heading for, all prediction schemes must observe their motion. This is done by means of certain pieces of information. We study how those pieces of information should be propagated in the network so that they “help” the prediction process as much as possible; by “help”, we mean bringing as much information as possible (in the sense of information theory). This leads to the novel concept of *entropy-based routing*, which is particularly suited to memory-limited devices such as those populating some wireless networks.

Part III: *Using* Mobility Prediction

We study to what extent a mobility prediction method can improve the quality of service of a cellular network. Prediction is used to reserve channels in neighbouring cells, and the chosen QoS metric is the percentage of blocked handovers. In order to remain prediction scheme-independent, we propose to use an *abstract prediction scheme model*, characterized by a few important parameters. We show that the channel reservation protocol can be abused by selfish mobiles, and thus propose an alternative *merit-based* protocol to alleviate this shortcoming.

Several routing protocols have been proposed for Disruption Tolerant Networks (*i.e.* ad hoc networks where packets must be delayed when no end-to-end path exists), but none makes use of the network predictability. We show how the predictability of the DTN nodes contact patterns can be mathematically described; we use this description to compute the end-to-end delay between any pair of nodes, leading to a framework that elegantly describes the behaviour of the DTN. Using this framework, we then build an original routing mechanism that seeks for a packet forwarding scheme that guarantees a given delay condition.

1.3 Related publications

This document is the result of more than four years of research that have led to several publications and presentations in various international conferences. The following list matches each technical chapter of this thesis with an enumeration of related articles we have written.

- **Chapter 3** (study of prediction in a WiFi network): *Delivery guarantees in predictable delay tolerant networks* (accepted at the *Networking'07* conference [FL07a]).
- **Chapter 4** (presentation of a generic mobility prediction scheme): *Learning movement patterns in mobile networks: a generic approach* (presented at the *European Wireless'04* conference [FLM04]) and *Évaluation d'une méthode de prédiction des déplacements de terminaux dans les réseaux mobiles* (presented at the *CFIP'03* conference [FLM03]).
- **Chapter 5** (on "entropy-based" routing): *Entropy-based knowledge spreading and application to mobility prediction* (presented at the *Co-Next'05* conference [FL05a]).
- **Chapter 6** (study of call admission in cellular networks): *Mobility prediction's influence on QoS in wireless networks: a study on a call admission algorithm* (presented at the *WiOpt'05* conference [FL05b]) and *Étude de mécanismes de prédiction de mouvement dans les réseaux mobiles* (master thesis [Fra04]).
- **Chapter 7** (routing in a disruption tolerant network): *Predictable Disruption Tolerant Networks and Delivery Guarantees* (accepted at *Networking'07* [FL07b], extended version on the *arXiv* library [FL06a]).

Those articles have been partially supported by the Walloon Region in the framework of the WDU programme (ARTHUR project), by the Belgian Science

Policy in the framework of the IAP programme (MOTION P5/11 project) and by the European E-Next Network of Excellence.

1.4 Dissertation outline

Chapter 2 is an overview of the prediction schemes that have been proposed in the literature. We not only focus on the prediction schemes themselves, but also on the way they can be used to improve the overall quality of service of the network.

Chapters 3 to 7 are devoted to our contributions. They are divided into three parts. The first aims at measuring how predictable wireless networks are, the second at finding ways to perform mobility prediction, and the last at inspecting how knowing a MH's future AP(s) allows one to improve the network QoS.

The last chapter finally gives a synthesis of this thesis and discusses along which lines this work could be followed.

2

State of the Art

Science is like sex: sometimes something useful comes out,
but that is not the reason we are doing it.

—RICHARD FEYNMAN

This chapter summarizes the ideas that have been proposed in the scientific literature to tackle mobility prediction. We divide it into two parts: the first presents ways to guess a terminal's future access point (or access router), the second how this information can help improve the network's performance[1].

Both parts begin with a presentation of mobility prediction-related (possibly experimental) industrial standards. This allows one to better grasp the differences between current needs and tomorrow's applications forecast by the research community.

2.1 Performing Mobility Prediction

2.1.1 Candidate Access Router Discovery

We first present a simple, experimental mobility prediction related protocol named *Candidate Access Router Discovery* (CARD). It has been described in RFC 4066 ([LSC$^+$05]); it has thus already been tested ([MN02]) and is backed by the industry.

The IETF has setup a group (mainly) responsible for studying how the impact of handovers could be mitigated, the SeaMoby (for *Sea*mless *Mob*ility) Working Group. This group states ([TKCK02]) that performing seamless handovers requires to know the capabilities of the surrounding access routers; this allows one to anticipate, for example, that the only way to guarantee the quality of an ongoing video streaming is to perform a vertical handover (*e.g.* from WLAN to 3G) since

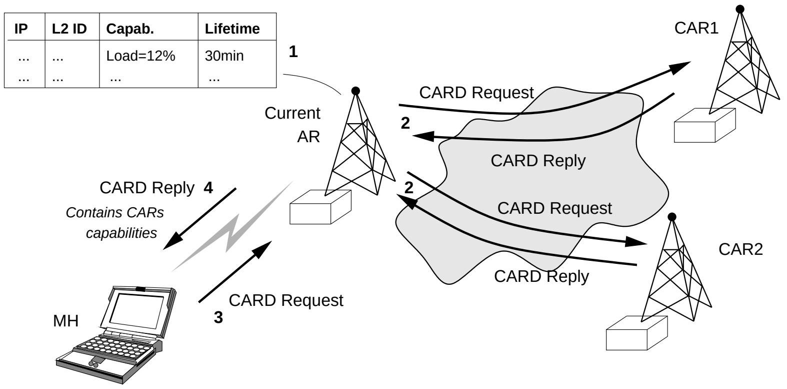
[1] Here, "performance" is a generic term with no specific meaning. It could mean reducing the packet loss as well as reducing the network operational costs.

the surrounding access points of the same technology are overloaded. Discovering the capability of the nearby ARs is thus the (main) aim of the protocol described below. The choice of the best next AR among all the suitable *Candidate Access Routers* (CARs) is not in the scope of this protocol.

Each access router maintains a “CAR table” summarizing the capabilities of each neighbouring AR. Those capabilities might be, for example, the current load of the subnet managed by the AR, or the accepted authentication methods. Some capabilities are only valid for a given amount of time; obsolete capability data are refreshed to keep the CAR table up-to-date.

Once a mobile host wants to perform a handover, it sends a request to its current AR asking *(a)* the capabilities of a specific CAR (discovered using a layer 2 method and identified by its MAC address, as explained section 1.1.2 in the case of WiFi), or *(b)* the capabilities of all the neighbouring CARs, or only those fulfilling certain requirements. The current AR replies using the pieces of information found in the CAR table.

The protocol is summarized in figure 2.1. This figure shows a direct message exchange between access routers; in practice, the required information could also be fetched from a centralized server.



IP	L2 ID	Capab.	Lifetime
...	...	Load=12%	30min
...	...	...	...

Figure 2.1: The *Candidate Access Router Discovery* protocol. Each AR maintains a *CAR table* that contains the known neighbouring CARs and their capabilities (**1**). Certain capabilities may need to be refreshed, in which case they are updated *via* a CARD Request/Reply exchange with the concerned CAR (**2**). Once the MH wants information about the surrounding CARs, it solicits its current AR (**3**) which replies using the CAR table’s items (**4**).

The messages transmitted between the MH and the current AR are sent using ICMP. Because of scalability concerns, transmitting packets between ARs is more problematic. The protocol standard proposes to use the *Stream Control Transport Protocol* (SCTP, see [SXM$^+$00]) which supports congestion control, fragmentation, and partial retransmission based on a programmable retransmission timer.

The content of the CAR table can be configured statically or learned using the motion of the mobiles themselves. The idea is that the table is initially empty, and is populated each time a mobile joins a new AR. It then sends a message containing the identity of its current access router to the previous one; this identity is added to the table.

2.1.2 The S-MIP architecture

We now turn to protocols and architectures backed by the research community.

Mobility prediction often involves scrutinizing how mobile devices physically move. In that sense, the *Seamless handoff architecture for Mobile IP* (S-MIP, [HZS03]) is quite typical.

In S-MIP, the handover is initiated by the MH which observes a poor connectivity. It informs a special network entity, the *Decision Engine* (DE), that it should change its point of attachment. The exact MH's next AR is determined by the DE, based on information provided by both the candidate ARs and the mobile itself:

- the mobile regularly (every second) sends the signal strength received by neighbouring ARs to the DE;
- the ARs regularly report the identity of the mobiles they are associated with.

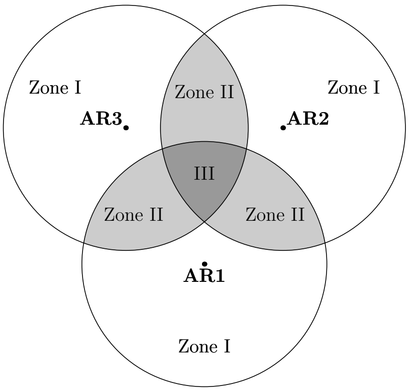


Figure 2.2: Localization using the S-MIP architecture.

The reported signal strengths are used to determine the location of the MH using a triangulation technique; more specifically, the MH is supposed located at the intersection of circles centered at ARs and whose radius is given by signal strength measurements. This is done under the assumption that the function linking antenna distance to signal strength is a negative logarithm (*i.e.* propagation in open-space or large indoor environments). The localization method only works if the access points are disposed in bee's nest (figure 2.2):

- *Zone I*: The MH is not likely to perform a handover since it only has contacts with one AR.

- *Zone III*: The MH localization can easily be found using a triangulation based on the three received signal strength.
- *Zone II*: A triangulation cannot easily be done since only two ARs are reachable: the MH is thus known to be located at the intersection of two circles, but two circles intersect at two distinct points. S-MIP continues however to work quite reliably since it can be shown ([HZS03]) that those two points must be next to one another (or one of them can be discarded).

The authors claim a precision of 2m using 802.11b access points ([ZCCS02]).

Once the MH location has been determined, the DE tries to see if the mobile *(a)* should not perform a handover, *(b)* is moving linearly towards another AR and should thus perform a handover, *(c)* is moving stochastically near the border of the zone covered by the current access point. The point of determining if the mobile behaves more like (a), (b) or (c) is that counter-measures are likely to be different for each case: if the mobile moves linearly, the handover can take place, but if it moves stochastically, the next AR cannot reliably be determined yet, and the mobile should stay connected to more than one AR (see section 2.2.2).

In order to determine the motion pattern of the mobile, its motion direction (North, West, South-East,...) is observed using the difference between the MH localizations at two different time intervals. If the DE detects a repeated direction of movement, it classifies the MH motion as linear; if the motion is neither linear nor stationary, it is considered to represent a stochastic movement (see figure 2.3).

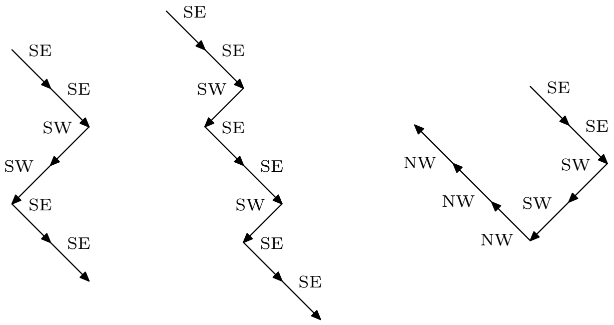


Figure 2.3: Various motion patterns. *From left to right:* ping-pong, linear motion, stochastic motion.

2.1.3 Tracking mobiles using sequence analysis

The sequence of cells (or APs/ARs) crossed by each mobile is an interesting source of information. It allows one to find repeated motion patterns that exhibit a certain regularity: MH movements never are totally random since they are constrained (*e.g.* the trajectory of cars depend on roads topology) and since they usually have a precise purpose (*e.g.* going to work). Extracting this motion regularity is a key element towards mobility prediction, since regularity rhyme with predictability.

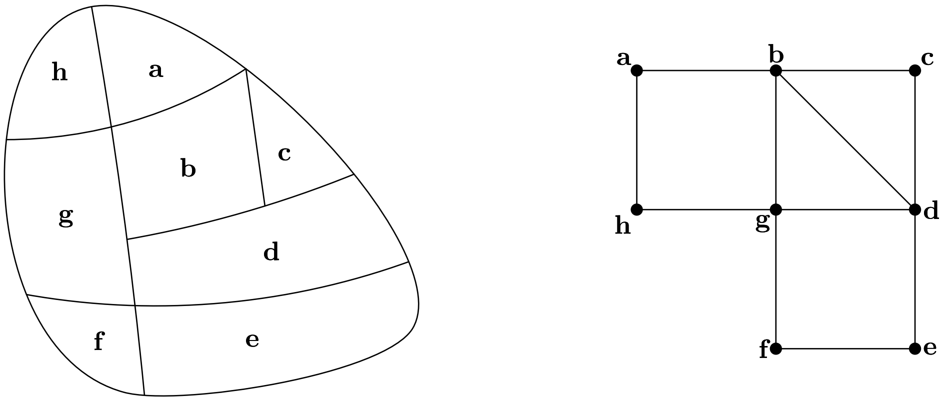


Figure 2.4: Each cell is modeled as an alphabet symbol. The path followed by a mobile moving from one cell to another can be represented as a sequence of symbols. The right part of the picture depicts the neighbouring graph: two cells are connected by an edge if and only if they have a common frontier. Notice that cells could be clustered to get a coarse-grained model.

Movement prediction using the analysis of typical trajectories represented as sequences of cells (or APs/ARs) has already been studied quite extensively ([LM95, AZ01, YL02, MRD04, Laa05, CS99, CZS98, CLSS00]). We here present (part of) a work from Bhattacharya and Das based on an information-theoretic approach ([BD99, BD02]).

The movement of mobiles moving in a cellular network is modeled as a series of cell ids. Each cell id is represented as a symbol belonging to a finite alphabet ϑ. The path followed by a mobile is thus modeled as a string of such symbols, called *movement history* (figure 2.4).

Notice that the consecutive symbols of movement history need not be distinct: it could be decided that the mobile position is recorded every half-hour, potentially leading to repeated symbols when the mobile does not move during a period of time longer than 30 minutes. Other methods can be used, such as updating the history every time the mobile changes its current cell (or every two cells), or mixing those two approaches (updating both every half-hour and every time the cell changes).

In what follows, the mobility model is a stochastic process $V = \{\mathcal{V}_i\}$ such that $\mathcal{V}_i$ assumes the value $v_i \in \vartheta$ if the i-th cell crossed is reported to be v_i. This stochastic process is supposed to be *stationary*, *i.e.* invariant to time shifts:

$$P(\mathcal{V}_1 = v_1, \mathcal{V}_2 = v_2, \ldots, \mathcal{V}_n = v_n) = \\ P(\mathcal{V}_{1+s} = v_1, \mathcal{V}_{2+s} = v_2, \ldots, \mathcal{V}_{n+s} = v_n) \quad (2.1)$$

for every shift s and for all $v_i \in \vartheta$ (the notation $P(\mathcal{V} = v)$ denotes the probability that the variate $\mathcal{V}$ takes the value v). The question is now to build a universal predictor or estimator for this process.

One common, simple (yet effective, see [SKJH04a]) model imposes the Markov assumption on the stochastic process. More precisely, the order-k Markov hypothesis imposes the following restrictions on the variates of V: for all $v \in \vartheta$:

$$P(\mathcal{V}_{n+1} = v \mid \mathcal{V}_1 = v_1, \mathcal{V}_2 = v_2, \ldots, \mathcal{V}_n = v_n) = \\ P(\mathcal{V}_{n+1} = v \mid \mathcal{V}_{n+1-k} = v_{n+1-k}, \ldots, \mathcal{V}_n = v_n) \quad (2.2)$$

where $P(A \mid B)$ denotes the conditional probability of A given B.

The parameters of Markovian models can easily be learned (more on this issue in section 3.2.2) using movement histories.

A natural question is to estimate the performance of such models and, for instance, study the impact of the model order. An elegant way of doing so is to compute the *entropy* (in the sense of information theory) associated with each model.

We first introduce basic definitions.

Definition 2.1. *The* entropy $H(\mathcal{X})$ *of a discrete random variable* $\mathcal{X}$ *that takes its values in the set* X*:*

$$H(\mathcal{X}) = -\sum_{x \in X} P(\mathcal{X} = x) \, \log_b P(\mathcal{X} = x)$$

One can arbitrarily specify the basis of the logarithm. As it is commonly done, we choose $b = 2$*; the entropy is therefore measured in* bits. *Notice that* $H(\mathcal{X}) \geq 0$.

In what follows, we take the liberty to write $P(x)$ instead of $P(\mathcal{X} = x)$ when there is no ambiguity.

Definition 2.2. *The* conditional entropy $H(\mathcal{X} \mid \mathcal{Y})$ *of a pair of discrete random variables* $\mathcal{X}$ *and* $\mathcal{Y}$ *is defined by:*

$$\begin{aligned} H(\mathcal{X} \mid \mathcal{Y}) &= -\sum_{x \in X} P(\mathcal{X} = x) \, H(\mathcal{Y} \mid \mathcal{X} = x) \\ &= -\sum_{x \in X} P(\mathcal{X} = x) \sum_{y \in Y} P(\mathcal{Y} = y \mid \mathcal{X} = x) \, \log P(\mathcal{Y} = y \mid \mathcal{X} = x) \end{aligned}$$

We can now give a sound measure of the entropy of the stochastic process $V = \{\mathcal{V}_i\}$.

Definition 2.3. *The* conditional entropy rate $H'(V)$ *is defined by:*

$$H'(V) = \lim_{n \to \infty} H(\mathcal{V}_n \mid \mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_{n-1})$$

The interested reader can refer to [vdL97] for an introduction to information theory.

An order-0 Markov model simply attributes a probability to each symbol. We thus have $P(v_n \mid v_1, v_2, \dots, v_n) = P(v_n)$ and

$$H(V) = H'(V) = -\sum_{v \in \vartheta} P(v)\ \log P(v) \tag{2.3}$$

Under the order-1 Markov assumption, the $V = \{\mathcal{V}_i\}$ process forms a so-called *Markov chain*, meaning that $P(v_n \mid v_1, v_2, \dots, v_n) = P(v_n \mid v_{n-1})$. Denoting $P(v_i \mid v_j) = P_{i,j}$, and according to definition 2.3, we have:

$$H'(V) = -\sum_i P(v_i) \Big(\sum_j P_{i,j}\ \log P_{i,j} \Big) \tag{2.4}$$

The probability $P_{i,j}$ is the probability of going to cell i from cell j; given a movement history, this can be estimated using a simple counting. The $P(v_i)$ values are the probability of being in cell i; they can be estimated as the number of times cell i appears in the movement history divided by the history length.

Alternatively, the $P(v_i)$ values are also the components of the vector $\boldsymbol{P}$ solution of $\boldsymbol{\Pi} \times \boldsymbol{P} = \boldsymbol{P}$, where $\boldsymbol{\Pi}$ is the transition matrix of the Markov chain (*i.e.* a matrix such that $\boldsymbol{\Pi}_{i,j} = \boldsymbol{P}_{i,j}$). $\boldsymbol{P}$ is the *steady-state vector*, which contains the fraction of time that a random process changing its state according to the transition matrix stays in a given state.

Equations (2.3) and (2.4) allows one to compute the entropy of order 0 and 1 markovian models; they can be generalized to higher orders. A simple example presented in [BD99] shows that the entropy of the motion process of a mobile that can possibly roam between 8 cells (thus *a priori* characterized by an entropy of 3 bits) is equal to 1.7 bits with order 0 and 1.2 bits with order 1 models. The marginal improvement of increasing the model order dies out soon; intuitively, the largest meaningful order is directly linked to the longest chain of dependency that can be found in the movement history.

Notice the importance of such measurements in the context of mobility prediction: the entropy estimates how predictable the MH is. Furthermore, markovian models can help predict a mobile's next cell; for example, according to (2.4), the most likely next cell for a mobile currently in cell i is $\operatorname{argmax}_j P_{i,j}$.

The *LeZi-update* algorithm described in [BD99] proposes to go one step further and replace markovian models with the tries built by dictionary-based compression algorithms such as Ziv and Lempel's LZ78 ([ZL78]). The underlying idea is that good compressors usually are good predictors since easily guessable symbols can virtually be left implicit, even if this requires to inefficiently code uncommon symbol sequences.

The LZ78 algorithm divides the string $v_1, v_2, \dots, v_n$ ($v_i \in \vartheta$) into $c(n)$ words $w_1, w_2, \dots, w_{c(n)}$. Those words are such that:

$$\forall j \geq 1,\ \exists\, i \geq 1,\ \exists\, s \in \vartheta : \quad i < j \quad \text{and} \quad w_j = w_i s \tag{2.5}$$

Less formally, the longest prefix of every word (the whole word but the last symbol) is equal to a previous word. To encode a word w_j, it is only required to encode *(a)* a reference to w_i (that hopefully requires less information than the direct coding of w_i) and *(b)* the symbol s.

For example, the movement history *"aaababbbbbaabccddcbaaaa"* would be parsed *"a,aa,b,ab,bb,bba,abc,c,d,dc,ba,aaa"* where the commas indicates the separation between words. Those words are added to a so-called dictionary often represented as a trie; see figure 2.5. Notice that this is not exactly the trie used by the LeZi-update algorithm; refer to [BD99] for more details.

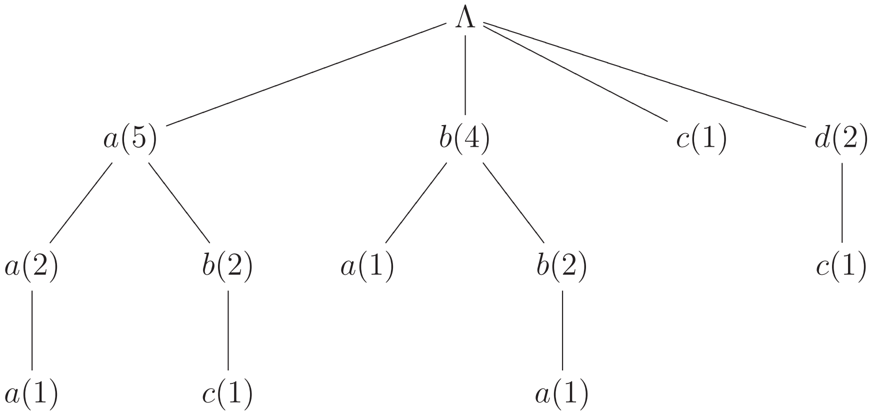


Figure 2.5: The trie generated by the Lempel-Ziv algorithm on the string *"aaababbbbbaabccddcbaaaa"*. The symbol Λ represents the empty word. This figure is inspired by [BD99].

The trie is annotated with the statistics of how many times a context has been explored. For example, the symbol a is *a priori* the most likely, with a probability of $5/(5+4+1+2)$. If the previous symbols read so far were a and b, the next symbol should be c. Notice that the next symbols can be predicted more precisely using the *prediction by partial match* (PPM); see [BD02] for more details.

As with Markov models, the LZ trie allows one to perform prediction of the next symbol or cell. Contrary to markovian models, LZ tries automatically adapt to the optimal context length.

Other works ([LM95, MRD04, YL02]) are based on the analysis of strings. An interesting example ([Laa05]) is the clustering of the paths frequently followed by mobile terminals; this requires to define a *distance measure* between paths. The problem can be elegantly solved if paths are represented as strings: the distance metric can then be defined as the *edit distance*, *i.e.* the minimum number of insertion, deletion, or substitution of a single symbol required to transform one string to the other (this distance is also used in [LBC98] for example).

2.1.4 Motion Prediction using Automatic Learning

Automatic/Machine Learning (ML[2]) and data mining are sometimes used to improve the behaviour of computer networks.

Such *Artificial Intelligence* (AI) related methods have been considered to tackle varied problems such as routing ([WKMT00]), saving the energy of small wireless devices ([YAG05]), improving the congestion control of TCP ([KGL05]), and determining the location of a terminal ([BB05]).

Some argue that the limitations of the Internet of today should not be addressed by the classical, algorithmic approach but rather by the tools of AI and cognitive systems. This yields to the concept of *knowledge plane* ([CPRW03, DL03]), whose goal is to maintain a high-level view of the applications requirements and give advices to the lower-level network components so as to fulfill them.

One way to perform mobility prediction is to extract the regular motion patterns from mobility traces. This allows one to compare the current trajectory of a given mobile to those patterns, and then infer the end of the trajectory; extracting regular patterns from noisy traces very naturally leads to AI-oriented techniques. The reader should be aware that the whole method is however based on two assumptions:

- Mobile movements exhibit a certain regularity. We have already discussed this hypothesis, and argued that it is reasonable;
- If the mobile trajectory is defined as its *physical* movement (as opposed to its path expressed as a series of access points), predicting the end of the trajectory does not directly solve the mobility prediction problem: the mobile location must be mapped to an AP id. Under certain circumstances, this mapping may not be trivial. For example, the cells of a cellular network generally have a large overlap and the mapping is thus not one-to-one; furthermore, the cell geometry must be known.

Several articles show that AI helps tackle mobility prediction ([CB04, BEJ97, HM02, HM99, ESE$^+$01, SK05]). We here focus on a data mining method presented by Yavaş et al. in [YKUM05].

We suppose that a large number of mobility traces has been collected. Each trace is an finite, ordered series of access points $\langle ap_1, ap_2, \ldots, ap_k \rangle$ that correspond to one movement of the mobile. These traces could for example represent the motion of a GSM in a cellular network; we would then consider that a movement ends (and thus the next movement begins) once the GSM stays in the same cell for a long time. Let D be the set of all the collected mobility traces.

Algorithm 2.1 gives an high-level overview of the regular pattern extraction method. The k-th run of the while loop finds the regular patterns of length k; the algorithm stops when k is too high and no pattern can be found. The main idea is

[2] A good overview of machine learning can be found in [Mit97].

Algorithm 2.1: An algorithm for finding regular patterns

```
1  $C_1 \leftarrow$ *all the cells included in* $D$;
2  $k \leftarrow 1$;
3  $L \leftarrow \emptyset$;
4  while $C_k \neq \emptyset$ do
5      forall $a \in D$ do
6          forall $s \in C_k$ with $s$ subsequence of $a$ do
7              // Add distance($a, s$) to the support for $s$;
8              // Remove from $C_k$ the elements with a
9              //         support smaller than $\texttt{supp}_{\texttt{min}}$;
10             $L \leftarrow L \cup C_k$;
11             $C_{k+1} \leftarrow$ candidateGeneration($L_k$);
12             $k \leftarrow k + 1$;
13         end
14     end
15 end
```

to create a set of candidate patterns of length k (initialized line 2), to remove the candidates that do not appear frequently in the mobility trace set D (line 9), and, using the candidates that have not been filtered out, build pattern candidates of length $k + 1$ (line 11).

The candidate patterns of length 1 are simply all the cells that appear at least once in the mobility trace set D.

The question now is: how can we build length $k + 1$ candidate patterns set C_{k+1} using length k candidates set C_k? We first find which candidate patterns are "popular" enough and discard the others. To do that, we consider all the (a, s) pairs with $a \in D$, $s \in C_k$ and such that s is a subsequence of a (*i.e.* the cells of s appear in a in the same order, potentially with other cells in between since the mobile movement is not entirely deterministic); a than gives a certain *support* (*i.e.* a real number) which is inversely proportional to the distance between a and s (line 7; this distance is basically implemented as the *edit distance* introduced page 20). If D contains a lot of patterns similar to s, it receives a lot of support; if the accumulated support is smaller than a chosen threshold (line 9), the candidate pattern is removed from the candidate pattern set.

Once all the unpopular candidate patterns have been filtered out, the C_k set is used to build the length $k + 1$ candidate pattern set C_{k+1}. This is done by considering all the elements $s \in C_k$; s is a list a cells denoted $\langle s_1, s_2, \ldots, s_k \rangle$. C_{k+1} is the union of all the sequences $\langle s_1, s_2, \ldots, s_k, s_n \rangle$ where the cells s_k and s_n have a common frontier (*i.e.* mobiles can move directly from s_k to s_n). The method thus requires to know the neighbouring relation between cells; this information can however be extracted from the motion traces.

Now that we have seen how the regular motion patterns can be found, we briefly explain how they can be used to perform next access point prediction. Each motion pattern $\langle s_1, s_2, \ldots, s_k \rangle$ can be divided into *mobility rules*:

$$\langle s_1 \rangle \rightarrow \langle s_2, \ldots, s_k \rangle, \quad \langle s_1, s_2 \rangle \rightarrow \langle s_3, \ldots, s_k \rangle, \quad \ldots \quad \langle s_1, \ldots, s_{k-1} \rangle \rightarrow \langle s_k \rangle$$

A *support* can be computed for each such mobility rule thanks to the support values computed previously for each mobility pattern (details can be found in [YKUM05]). If a mobile has followed a path $\langle t_1, t_2, \ldots, t_i \rangle$ until now, this prefix can be compared to the "head" (left-hand) of each mobility rule. Once a match is found, the first access point reported in the mobility rule's "tail" is a potential next access point whose probability is proportional to the mobility rule's support value.

2.1.5 Location-based path prediction

Knowing the location and velocity of a mobile obviously helps guess where it is heading for. The problem is thus *(a)* to monitor the mobile's current (physical) trajectory, *(b)* to extrapolate this trajectory, and *(c)* to match the extrapolated trajectory with the next access point. Item *(a)* can be addressed using GPS devices which are now both cheap and common ([TYK05]), or some form of positioning method based on multilateration (*i.e.* measuring the signal's *Time Difference of Arrival* (TDOA) at the neighbouring BSs) or triangulation ([Fra, CS98a]). Item *(b)* requires building a model of the motion, as we will see below. Item *(c)* has already been discussed in the previous section.

Several articles have proposed to perform mobility prediction using some form of trajectory extrapolation ([LH03, ZM05]). The simplest form of extrapolation is of course the linear approximation; it has been used for example to estimate the duration of links between the nodes of a Mobile Ad Hoc Network (MANET, see [SLG01]). Since such simple models are not always satisfactory, some works ([LBC98, ZM05]) rely on more advanced methods such as *Kalman filters* (the mobile is then supposed to have a (continuous) state and a linear state-to-state transition function; a Markov hypothesis is made, and measurement uncertainties are supposed gaussian[3]).

In this section, we give an overview of Soh and Kim's hybrid method for cellular networks based on location information and road topology ([SK03, SK04]). In those articles, the central assumption is that the mobile terminal is a vehicle that is able to regularly send its physical location to the base station (*e.g.* every 1s).

Each BS maintains a database of the roads within its coverage area (this information can be extracted from a digital map service). Each road is supposed linear (bends are approximated by linear segments). The BS records the average time

[3] The interested reader can refer to [FP03] for an good introduction to the method.

taken to transit each road, the probability of transition from one segment to the next (this transition is modeled as a second-order Markov process, see figure 2.6).

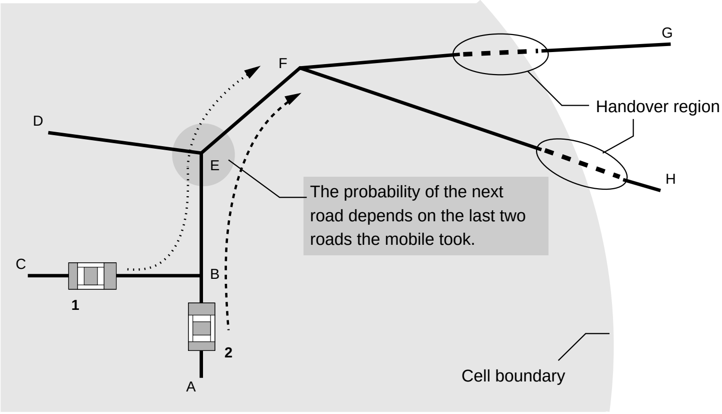

G
F
Handover region
D
E
H
The probability of the next road depends on the last two roads the mobile took.
C
B
1
2
A
Cell boundary

Figure 2.6: Prediction using road topology information. Once they arrive at the E junction, the BS computes the probabilities that terminals 1 and 2 continue their way towards D or F. Since the last two roads they followed are different ($\langle CB, BE\rangle$ and $\langle AB, BE\rangle$ respectively), those probabilities are estimated independently.

The handovers of all mobiles are monitored continuously, so the roads where handovers usually appear can be flagged. This allows one to automatically detect the boundaries of the cell instead of assuming an hexagonal shape.

Using those pieces of information, the likely path of a mobile can be estimated in advance, so both the handover probability and the time remaining before this handover can easily be derived.

2.2 Using Mobility Prediction

The first section considers a standardized protocol involving mobility prediction that has already been deployed or, at least, tested in labs. It could be largely used in the future, depending of its experienced utility and efficiency. The protocol presented in this section has been normalized by means of IETF RFCs ([IET]).

We then present other ideas that have been proposed by the research community.

2.2.1 Mobile IPv4 Fast Handovers

We here briefly describe the protocol aimed at providing fast handovers for Mobile IP is described in [KP06][4]. This protocol can use the knowledge of the next AR to perform packet relocation as explained below.

[4] This protocol is adapted from the equivalent protocol defined for Mobile IPv6, see IETF RFC 4068 ([Koo05]). It is similar to the proposition of Calhoun et al. ([Cal00]).

A MH is first supposed to guess the IP address of its *next access router*, NAR; the *previous AR* (which gives the MH its connectivity before the handover) is denoted PAR. The CoAs given by the PAR and NAR are denoted PCoA and NCoA (respectively). The exact means by which NAR is identified is not part of the *fast handover* protocol, but one can reasonably assume that currently layer 2 triggers are the preferred way to find this piece of information. We have already seen how that can be performed for WiFi networks in section 1.1.2. The protocol performance will typically be enhanced by using the appropriate triggers.

If only the layer 2 identification of NAR is known, special messages can be exchanged with the PAR in order to gain knowledge of data such as its IP address and the subnet it is responsible for: this *next-AR discovery phase* is depicted in figure 2.7. The NCoA of the mobile is temporarily chosen to be that of the NAR: the FA is not co-located, otherwise DHCP (for example) would be used and would cause long delays.

Notice that, using IPv6, the NCoA can be chosen right away thanks to the knowledge of the NAR subnet prefix by means of *address auto-configuration* (see RFC 2462, [TN98]).

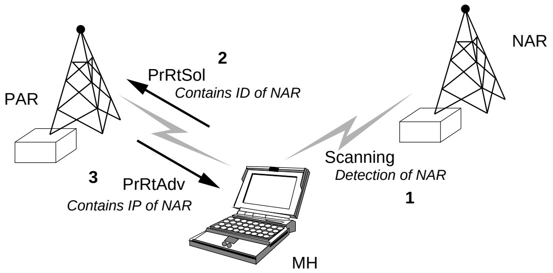


Figure 2.7: The PrRtSol/PrRtAdv messages. A layer 2 mechanism allows the MH to find the layer 2 address of the next access router (**1**). The MH sends this address to the PAR thanks to the *Proxy Router Solicitation* (PrRtSol) message (**2**). PAR sends a *Proxy Router Advertisement* in response, containing various pieces of information related to NAR (**3**).

The second part of the protocol begins when the MH is ready to perform its handover; it is depicted in figure 2.8. The mobile first sends a special *Fast Binding Update* (FBU) message to PAR; the PAR replies with a *Fast Binding Acknowledgment* (FBack). This message informs the PAR of the IP address of the NAR, and triggers the PAR to setup a tunnel between PAR and NAR: PAR sends a *Handover Initiate* (HI) message to NAR (containing the parameters of the tunnel and security/authentication information), and the NAR replies with an *Handover Acknowledge* (HAck) message.

Once the tunnel is setup, packets received by PAR for the MH are tunneled to the new access router, which buffers them. The mobile can then perform its handover.

Once the mobile is connected to the new link, it initiates a FBU/FBack exchange

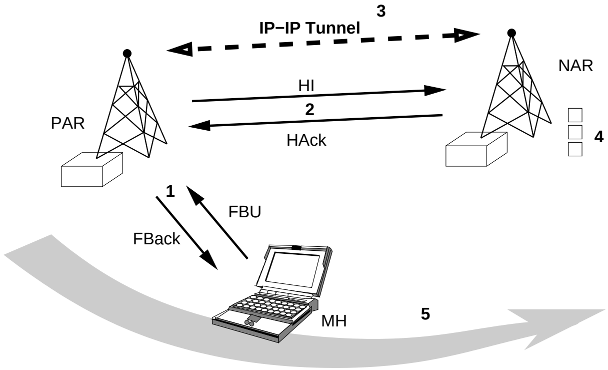


Figure 2.8: The Fast Handover Protocol: before handover. The MH and PAR exchange FBU and FBack messages (**1**) in order to initiate the handover; the FBU contains the IP address of the NAR foreseen by the MH. The PAR contacts the NAR using a Handover Initiate; the NAR responds with an acknowledgment (**2**). PAR and NAR then setup a bidirectional tunnel; the packets destined for the MH are tunneled by PAR and buffered by NAR. The MH is now free to perform the actual handover (**5**).

with the NAR, which can then finally forward the packets it had buffered. The fast handover protocol ensures that the MH continues to receive and send packets thanks to the (bi-directional) tunnel before the NCoA has been registered with the mobile's home agent: the registration latency is thus disengaged. Figure 2.9 represents this part of the protocol.

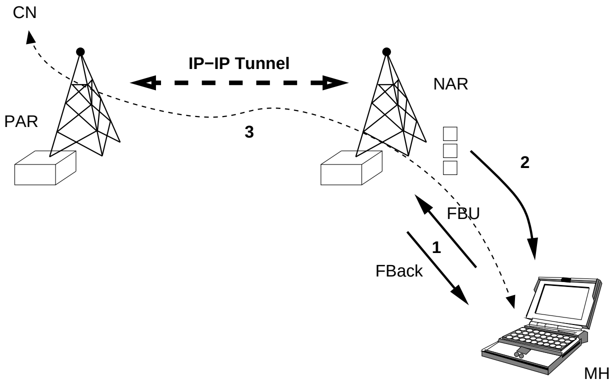


Figure 2.9: The Fast Handover Protocol: after handover. Once the new link is setup, the MH registers with the NAR using a FBU/FBack exchange (**1**). The buffered packets can be delivered (**2**). The MH still can exchange packets with the CN using the tunnel (**3**). This process continues until the MH registers its new CoA.

Notice that it is not clear whether the Fast Handover Protocol allows seamless handovers in practice. Simulations ([HSSEM02]) tend to show that this protocol used together with H-MIP (see section 1.1.3) allows one to reduce handover latency to about 300 milliseconds, far too much to allow, for example, an uninterrupted VoIP call.

2.2.2 The S-MIP architecture

In the rest of this chapter we present some ideas proposed by the research community.

We have already seen how the S-MIP architecture solves the mobility prediction problem (see section 2.1.2). We now show how it can be used to improve the network reliability.

S-MIP is built on hierarchical schemes (see the explanation of H-MIP in section 1.1.3) and fast-handover (see section 2.2.1 above). With S-MIP, as we have already mentioned, the handover is mobile-initiated, but network-determined; this means that the network's Decision Engine (DE) should try to unambiguously determine the MH's next AR. We have also pointed out that the DE regularly receives feedback from both MHs and ARs; it is thus capable of choosing to favour handover towards less congested cells (and thus perform load balancing).

Once the MH detects an imminent handover, it uses the fast handover protocol described above. The current AR sets tunnels to the candidate next ARs discovered by the MH. Those candidates send to the DE a message containing status information (*e.g.* current load) that could help determine the best next AR. Once it has received those messages and once the motion behaviour of the MH has been determined[5], the DE is asked to choose which candidate AR should be used as the next AR. If the mobile is moving linearly, the next AR can be determined unequivocally. If the MH is moving stochastically or stands in the overlapping zone between two ARs, multiple next ARs are elected. Those ARs should retain the MH binding in preparation for the MH return (in case of ping-ponging, *i.e.* an oscillation between two access points); this binding can be removed as soon as the DE has decided that the handover is terminated. The DE notifies the elected potential next ARs and asks the GFA to multicast the messages destined for the MH to all the ARs involved in the handover (the previous AR and the potential next ARs). The packets tunneled from the previous AR are flagged (using on option bit in the IP header) and buffered at the potential next ARs. The multicast packets are also buffered and flagged differently.

It can be shown that tunneled packets are likely to be chronologically older than multicast packets. The ARs are thus required to send them first. Packet multicasting is stopped once the decision engine has determined that the handover is terminated (*i.e.* potential ping-ponging is certainly finished).

[5] Recall that S-MIP mobility prediction engine classifies to motion of the mobile as moving linearly, moving stochastically or standing between two ARs (see section 2.1.2).

2.2.3 Improved paging using cell sequence compression

In a cellular network, an important problem is keeping track of each mobile location. This is critical since each time, for example, a GSM must be contacted to receive a new call, it must be searched in the whole network; this operation should be optimized so as to consume as little wireless bandwidth as possible.

In a GSM network, an idle mobile is required to update its position regularly by means of *update messages*. The reverse operation, *i.e.* finding the mobile current cell, is called *paging*.

This problem is commonly tackled with an expanding ring search: the mobile is first searched in the cell it has sent its last update message from. If it is not found, the paging is performed simultaneously in the surrounding cells. This search continues, involving more cells at each iteration, until the mobile is found or the procedure is considered too bandwidth-expensive.

With the LeZi-Update ([BD02]) algorithm, mobility prediction is proposed as a way to reduce paging costs, as explained below.

As we have seen in section 2.1.3 that the path followed by a mobile can be represented as a string of symbols. This string can be compressed using the Lempel-Ziv algorithm, yielding a series of words.

It is proposed that this series of words be encoded by the mobile and decoded by the base station; each update message contains a word (which corresponds to the mobile path). If the network wants to locate the mobile, it first uses the last word received as a context. Using the statistics included in the LZ dictionnary trie (remember the example given in figure 2.5), the most likely next update message words are guessed. The cells corresponding to those words can be paged with the most probable cell first.

2.2.4 Decreasing handoff blocking using prediction

In a cellular network, a cell can receive two different types of calls: handover calls and new calls. A user is likely to find the forced termination of an ongoing call more objectionable than the blocking of a new call. We should thus find countermeasures that mitigates the former at the expense of the latter.

This can be done by reserving resources in advance in the next cell reached by mobile phones ([LAN95, CB00, KKK01]). A certain amount of bandwidth (the *guard bandwidth*) can be pro-actively reserved to receive roaming mobiles (thus decreasing the handover blocking probability); the bandwidth available for new calls is therefore reduced, hence increasing the new call blocking probability.

The amount of reserved bandwidth can be fixed once and for all if we assume a stationnary traffic. If this hypothesis does not hold, the wireless bandwidth is either over or under-utilized; we thus advocate a more dynamic approach.

The method presented in [SK01] is a good example of such a dynamic, proactive reservation scheme targeted at cellular networks. It is summarized in what follows.

We assume that a mobility prediction algorithm is available; it sends a handover trigger when a fixed amount of time remains before the expected handover. This trigger contains an identifier of the predicted next BS.

Each mobile is supposed to use one class of service; each service class i is characterized by the bandwidth it requires, B_i. Since predictions might be wrong, and since the current call could end after the prediction trigger but before the actual handover, it is not necessary to reserve a bandwidth B_i at the expected next cell, but only a fraction of that amount. Each cell defines a $\boldsymbol{R}$ matrix so that, when a mobile using service class i performs a handover to cell j, the amount of bandwidth reserved in cell j is equal to $B_i\boldsymbol{R}_{i,j}$.

The $\boldsymbol{R}$ matrix is adapted by each cell so as to reach the required handover blocking probability (*i.e.* certain elements are increased if handovers are too likely to be blocked, decreased otherwise). This way, the $\boldsymbol{R}$ matrix should converge towards a value that allows us to reach the aimed blocking probability.

Each time a prediction trigger is received, the guard bandwidth of the predicted next cell is increased by $B_i\boldsymbol{R}_{i,j}$; it is decreased by the same amount as soon as the call is ended or the mobile performs its handover — possibly not in the expected cell.

The call admission procedure for new calls of service class i is quite straightforward. The current cell accepts to route the new call[6] if the spare bandwidth minus the guard bandwidth is higher than B_i.

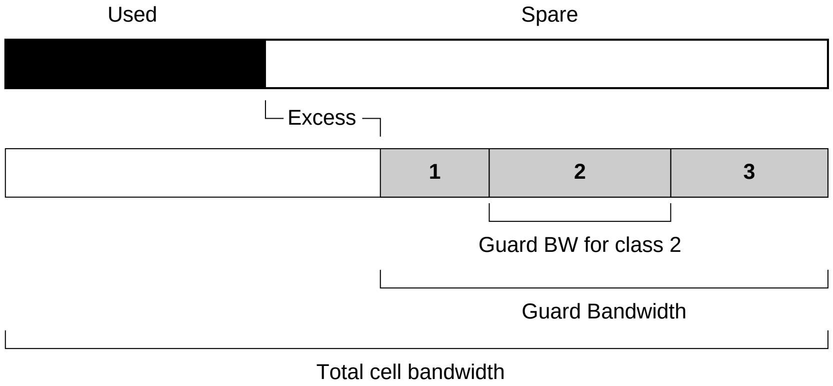


Figure 2.10: A call admission control procedure. Notice that there might be no "excess" bandwidth if the sum of the guard and the used bandwidth is greater than the cell capacity.

For handover calls, the procedure is a bit more complicated. Two cases must be

[6] In fact [SK01] also deals with the bandwidth requirements in the fixed network infrastructure, a problem omitted here for brevity.

distinguished (see figure 2.10); when a mobile using the class of service i performs a handover:

- Either the sum of the bandwidth used by current calls and the guard bandwidth is smaller than the cell capacity (there is some “excess” bandwidth, see figure 2.10). In this case, the call is admitted if the sum of the excess and the guard bandwidth reserved for class i is greater than the requested bandwidth B_i.

- Either it is not the case (the spare bandwidth is less than the guard bandwidth), in which case a proportion r of the spare bandwidth is assigned to the handover call (the call is rejected if this does not suffice). The ratio r is the proportion between the guard bandwidth reserved for class i and the total guard bandwidth. This rule provides a form of fairness between the different classes: without it, the traffic classes with small bandwidth requirements could benefit from the reservations done for other classes.

Part I

Mobility Prediction *Feasability*

3 Predictability of Wireless Networks

When we write programs that "learn",
it turns out we do and they don't.

—ALAN J. PERLIS, *Epigrams on Programming* (1982)

Before studying how mobility prediction could be performed in practice, we first try and determine to what extent the movements of mobile hosts are predictable. The evaluation of this degree of predictability is done using real mobility traces.

In this study, we emphasize the differences between the methods that predict the behaviour of each particular mobile and those that model the motion of a whole group of users. We find an unexpectedly small difference between both methods in terms of accuracy.

We also try and find which pieces of information (*e.g.* time of the day) help the prediction process.

3.1 Introduction

Mobility prediction methods can be classified into two main families:

- **MH-centric** (MHC): the agent performing predictions is bound to a MH; it builds a model of this particular MH's movements (*e.g.* [MRD04, Laa05, SK05, CS99, CZS98]);
- **AP-centric** (APC): the prediction agent is bound to an AP; this AP builds a model using the motion of the MHs passing by (*e.g.* [HM02, SK04, SK01]).

Those models can be built on-line, as the mobiles move, and can perform a prediction anytime.

As we have already mentioned in section 1.1.4, the pieces of information that allow one to deduce the likely motion of a mobile terminal are very varied (*e.g.* GPS coordinates). In what follows, we assume that the piece of information used is the sequence of most recently encountered APs. This assumption is not strong since it only requires MHs to record the last APs they have been associated with.

The various prediction methods presented in the literature have rarely been validated using mobility traces extracted from a real network. [SKJH04b] is a notable exception which shows that simple markovian models perform nearly as well as other more complex methods (see also [HKA04]). We thus focus on those models.

In the following, we present an (essentially quantitative) comparison between MH and AP-centric prediction methods using the mobility traces of a large-scale WiFi network. It is expected that the conclusions drawn here could be applied to other types of mobile networks.

3.2 Next-AP predictors

A *next-AP predictor*, or *prediction agent*, is the entity responsible for building a model of MH's movements; this model can be used for predictive purposes.

3.2.1 Centralized and decentralized methods

In this chapter, we study the differences between the two most popular prediction schemes.

In the first method, APC, the prediction agents are the APs. Each AP builds a model of the movements of mobiles passing by. The MHs' involvement is minimal, since they only send during each handover an identification of their previous APs. This architecture is particularly well suited to situations where predictions are mainly useful to the fixed network infrastructure (which could, for example, use it to reserve resources anticipatively). This scheme can quickly get an estimation of the model learnt since it improves every time a mobile goes by.

The second method, MHC, is more distributed: every MH builds a model using its own movements. It is expected to be more reliable since more specific: the behaviour of a particular mobile cannot be simplified to the mean behaviour of all the MHs moving in the same area. However, this scheme does not fit well the standard wireless network paradigm, where terminals are supposed to be small, memory and processing limited devices (such as a low-end GSM), and not suited to running a learning algorithm. Moreover, no prediction can be made when a MH visits APs for the first time.

3.2.2 Markovian models

We model MHs' motion habits thanks to their *location history* (or *trace*), *i.e.* the sequence of APs crossed during their journey. Considering each AP as a symbol of a (finite) alphabet, a MH's trace is a sequence of symbols and prediction aims at guessing symbol $i+1$ given the first i.

Observing MHs' motion allows a prediction agent to tune the model's parameters so that prediction improves over time

It has been shown ([SKJH04b, SKJH04a]) that in this context, simple Markov predictors perform as well as other, more complex methods[1] (such as [BD99, YL02, CW84, JSA00]). We thus only consider this class of predictors here.

Let $\mathcal{L} = \{L_1, L_2, L_3 \dots\}$ be the set of locations and $L = L_1, L_2, L_3 \dots$ a location history. The *order* n markovian hypothesis is:

$$P(L_i = l | L_1, \dots, L_{i-1}) = P(L_i = l | L_{i-n}, \dots, L_{i-1}) \qquad \forall\, l \in \mathcal{L},\, i > n \tag{3.1}$$

Less formally, this equation states that the stochastic variable that describes the next-AP probability follows a distribution that only depends on the last n symbols. We assume a stationary distribution[2].

The next-AP distribution can easily be learnt on-line. We assume that the agent responsible for building the markovian model is regularly notified of MH(s) movements.

If we denote L^m the location history of mobile m, the order-n model estimation rule is:

$$P(L_i = l \mid L_{i-n}, \dots, L_{i-1}) = \frac{\sum_{m \in \mathcal{M}} O(L^m_{i-n}, \dots, L^m_{i-1}, l\,;\, L^m)}{\sum_{m \in \mathcal{M}} O(L^m_{i-n}, \dots, L^m_{i-1}\,;\, L^m)} \tag{3.2}$$

where the $O(\cdot\,;\cdot)$ operator finds the number of occurrences of its first argument in its second, and $\mathcal{M}$ is the set of mobiles involved.

In the case of MHC, each terminal only models its own motion, thus $\mathcal{M}$ is a singleton.

When a model is used to perform a prediction, the most probable next AP (given the current *context*, *i.e.* the MH's last n APs) is chosen. No prediction can be performed if the context has never been observed before. To limit the consequences of this possibility, we build together with each order-n model, $n-1$ other models of order $n-1, n-2, \dots, 1$. If a prediction cannot be performed because the current context is seen for the first time, we fallback on a lower-order model.

We do not aim at predicting when a mobile will enter or leave the network; only the proper inter-AP movements are taken into account.

[1] To be fair, some of those not only aim at location prediction, but at other purposes such as mobile paging.

[2] This hypothesis is confirmed by the results presented in section 3.5.3.

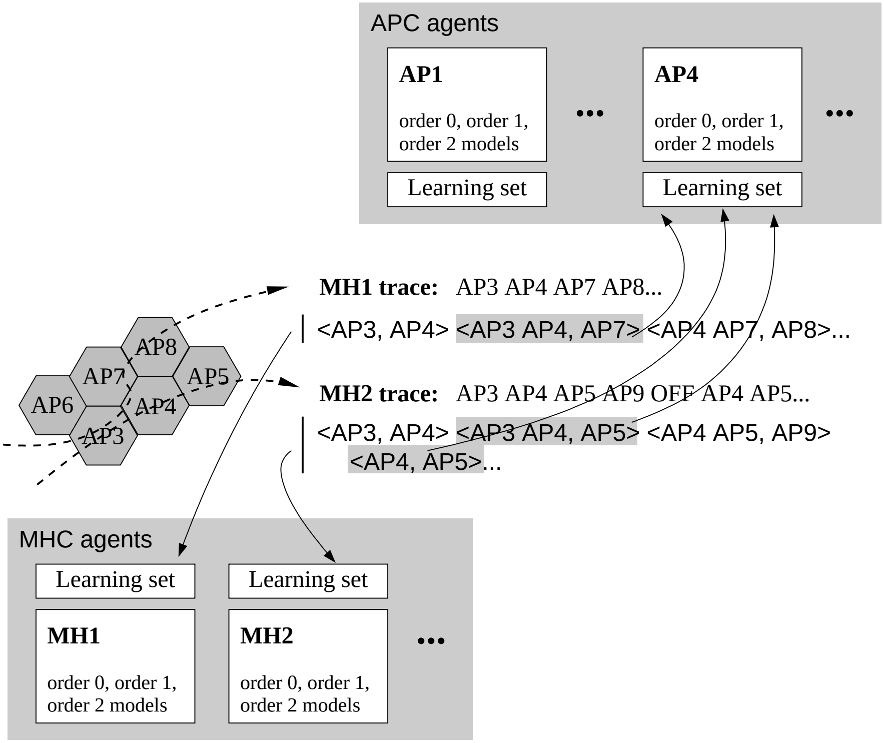

APC agents
AP1
order 0, order 1, order 2 models
Learning set
AP4
order 0, order 1, order 2 models
Learning set
MH1 trace: AP3 AP4 AP7 AP8...
<AP3, AP4> <AP3 AP4, AP7> <AP4 AP7, AP8>...
MH2 trace: AP3 AP4 AP5 AP9 OFF AP4 AP5...
<AP3, AP4> <AP3 AP4, AP5> <AP4 AP5, AP9> <AP4, AP5>...
AP8
AP7
AP5
AP6
AP4
AP3
MHC agents
Learning set
MH1
order 0, order 1, order 2 models
Learning set
MH2
order 0, order 1, order 2 models

Figure 3.1: Overview of the learning process for order-2 markovian models. The mobiles' motion generates location histories that allow agents to build a set of "context, next-AP" couples (here denoted between brackets). The special "OFF" AP is introduced when the MH leaves the network; since we do not aim at predicting this event, those APs do not appear in the learning sets.

3.3 Wireless traces

The traces used have been collected by Dartmouth University in the context of the CRAWDAD project ([Kot05]). It is a collection of events generated by the WiFi network of the campus. *Syslog* and *SNMP* data have been recorded for 2 years and cover 6202 MHs and 575 APs ([KE02]).

This data have been analysed ([SKJH04a]) to extract the actual movement traces (*i.e.* for each MH, a sequence of APs). A special AP, denoted *OFF*, indicates that a mobile has been 'deauthenticated' or that it has not generated any activity since at least 30 minutes; we then consider that it has been disconnected from the network.

Each MH is identified using its MAC address; we assume that each MAC address matches one (and only one) user. Apparatus with more than one interface and apparatus shared by more than one person are considered rare.

As it is commonly done in *machine learning*, the traces have been divided into two distinct parts: 5/6 are used as a learning set, the rest being an independent test set used solely in section 3.6.

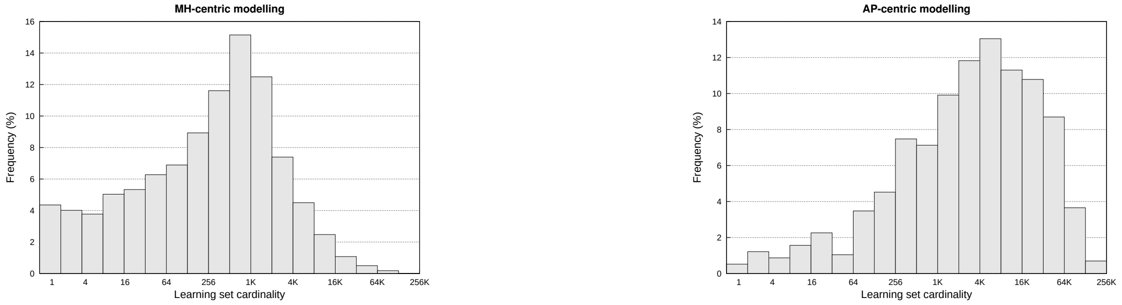


Figure 3.2: Distributions of learning sets cardinalities regarding MHC (left) and APC (right). The x-axis is logarithmic.

In the following, the same data are exploited to perform MH and AP-based predictions. More formally, each time MH m moves, its last n movements $L^m_{i-n}, \ldots, L^m_{i-1}$ and the next AP L^m_i are used to learn the parameters of an order-n markovian model; those “context, next AP” couples are the elements (or *learning samples*) of the learning set. With MHC, the prediction agent is bound to 'm'; with APC, it is bound to L^m_{i-1}. Notice that the contexts fed to L^m_{i-1} always end with L^m_{i-1}; an order n model built with APC is thus as complex as an order $n-1$ model built using MHC.

Figure 3.1 depicts graphically the learning process. Each time a terminal moves, its new AP is appended to its mobility trace; a special 'OFF' AP is added when the MH is disconnected from the network. This trace is then converted to a series of “context, next-AP” couples; the maximal context length depends on the order of the models built. The couples corresponding to MH m populate the learning set bound to m (left); the couples whose context ends with AP p are the learning samples that compose the learning set of p's agent (right).

Notice that the number of elements in a mobile trace is nearly equal to the number of “context, AP” couples it generates (but not exactly because of the 'OFF' APs), and that an agent performs a prediction each time it receives a new learning sample. Thus, the number of movements, the learning set cardinality, and the number of predictions are virtually equal and can be used interchangeably.

Figure 3.2 compares APC and MHC in terms of the distributions of the learning sets' cardinality (*i.e.* the number of “context, next AP” couples). A lot of MHs barely move: 12% perform less than 8 movements (plot on the left, sum of the percentages reported in the first 3 bars). On the contrary, few APs are unpopular: 15% are crossed by less than 256 MHs. The impact of under-learning should thus be more pronounced using the distributed MHC method.

3.3.1 Ping-ponging

It is known (*e.g.* [CLF06]) that such dataset exhibits the *ping-ponging* artefact (*ping-ponging* is defined as repeatedly changing one's current association back and forth between two —or more— access points).

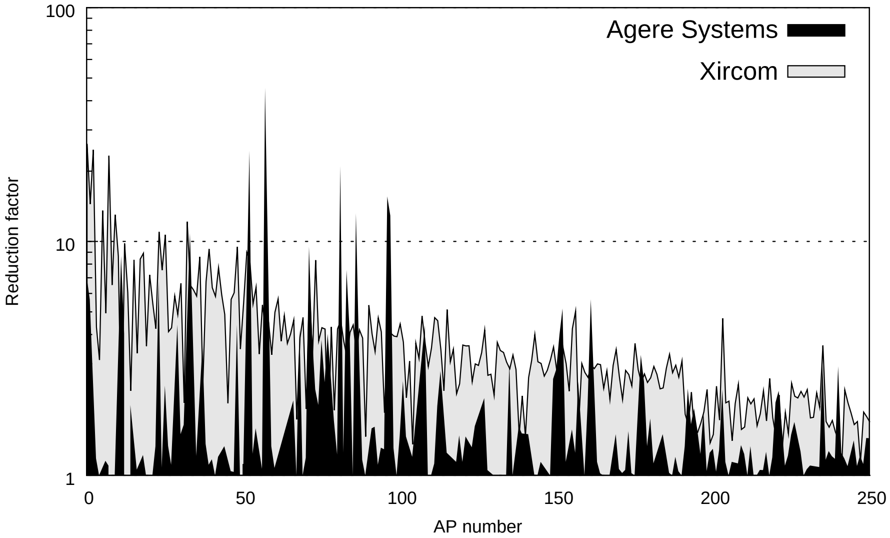


Figure 3.3: Ratio of learning sets cardinalities before and after ping-ponging removal for two network interface manufacturers. APs are sorted by decreasing mean reduction ratio.

A device usually chooses to connect with the reachable AP characterized by the strongest signal, which depends on the mobile's motion and —on a shorter time scale— on the signal propagation conditions. A (device-dependent) hysteresis mechanism usually decreases ping-ponging impact.

Mobility prediction is concerned about the physical movements of mobile terminals, not about those quick artefacts. If the purpose of prediction is, for example, to reserve an amount of bandwidth B, a simple strategy could be to allocate $B/2$ at both APs between which ping-ponging appears; those APs are thus considered as one entity. As a consequence, it is likely that the mechanism using prediction as an input would consider APs that frequently ping-pong as a single entity. This does not mean that predicting ping-ponging is not an interesting topic, but that it is only marginally related to the question studied here. We thus try to remove this artefact.

Considering the location history $L_1, \ldots, L_n$ of a MH, the movement to L_n is classified as ping-ponging if $L_{n-2} = L_n$. This simple rule surely does trigger "false positive": MHs physically moving back and forth from an AP to another are classified as ping-ponging. However, such situations are likely to be rare as confirmed by the results below.

Table 3.1 shows how frequent ping-ponging is, and figure 3.3 correlates it with two interface manufacturers. Table 3.1 gives the ratio between the models learning set cardinality with and without the elements considered as ping-ponging. We can draw several lessons from those numbers.

First, our ping-pong criterion looks reasonable since the results vary between manufacturers: it finds handovers triggered by a technological cause, not by actual MHs' movements. This is confirmed by table 3.2 which shows that prediction accuracy is virtually manufacturer-independent once ping-ponging has been removed.

Second, the reduction ratio of the learning sets cardinality caused by ping-ponging is high: about 3 movements out of 10 are removed. This is likely to explain some results presented in [SKJH04a], showing that it is useless to use complex methods (*e.g.* based on Lempel-Ziv): simple, low order markovian models suffice to predict ping-ponging. It would be interesting to repeat the experiments without this bias.

Third, since the MH's location history depends on interface manufacturers when ping-ponging is not removed, APC predictions could be slightly improved if different models were built for each one.

Finally, the standard deviations reported in table 3.1 are bringing to light very different profiles of MHs: some experience lots of ping-ponging, others very few. Furthermore, quite different results are found when considering all the MHs as equally important (table 3.1, left column) or when they are weighted according to their trace length (right column) since ping-ponging generates a lot of artificial movements.

From now on, all the ping-pong movements have been expunged from the traces.

The prediction accuracy for both APC and MHC is then given in table 3.2, which can be compared to table 3.3. The accuracy of MHC does now virtually not depend on the manufacturer and, for high-order models, the difference between APC and MHC is quite small (from 55.2% to 59.8%). One can conclude that ping-ponging is easily predictable since performance is now worst than reported in

	Learning set reduction ratio	
Manufacturer	Not weighted	Weighted
Agere Systems	44.9% (23.8%)	63.1% (27.3%)
Xircom	52.6% (27.4%)	65.4% (25.0%)
Aironet Wireless	58.9% (26.1%)	71.3% (26.0%)
Apple Computer	72.3% (20.9%)	79.0% (20.1%)

Table 3.1: Ratio of learning sets cardinalities by interface manufacturer. Standard deviations are given between parentheses. The numbers given in the left column consider that all the mobiles are equally important; in the right column, reduction ratios are weighted according the mobiles' traces length. Only manufacturers representing at least 200 MHs are shown.

Order	**APC**	**MHC**				
		Average	Agere	Xircom	Aironet	Apple
1	28.7%	39.4%	*38.8%*	*40.0%*	*39.9%*	*38.6%*
2	47.6%	58.4%	*56.6%*	*59.3%*	*59.3%*	*58.9%*
3	53.0%	59.5%	*57.4%*	*60.4%*	*60.3%*	*60.1%*
4	55.2%	59.8%	*57.5%*	*60.7%*	*60.7%*	*60.4%*
5	55.4%	59.6%	*57.5%*	*60.6%*	*60.5%*	*60.2%*

Table 3.2: Prediction accuracy without ping-pong movements. Using models of higher order (greater than 5) shows no improvement.

table 3.3

3.4 Next-AP predictions accuracy

Table 3.3 shows the next-AP prediction accuracy using both APC and MHC, for various model orders.

	W/o *ping-pong*		With *ping-pong*	
Order	**APC**	**MHC**	APC	MHC
0	**N/A**[3]	**24.0%**	N/A	38.7%
1	**28.7%**	**39.4%**	40.9%	68.4%
2	**47.6%**	**58.4%**	64.5%	72.9%
3	**53.0%**	**59.5%**	64.9%	73.0%
4	**55.2%**	**59.8%**	65.1%	73.0%
5	**55.4%**	**59.6%**	64.9%	72.8%

Table 3.3: This table gives the prediction scheme performance. Each number corresponds to the ratio between the number of accurate predictions and the total number of predictions. The columns on the right show the performance one would obtain if ping-ponging were not removed.

We first observe that a simple, statistical approach[4] gives unsatisfactory results. Models improve quickly: order-2 models are quasi-optimal. Beyond that, performance improves very slowly and reaches its apogee with order-4 models. Performance decreases with models of higher order, showing a slight over-learning. The columns on the right are those obtained if the ping-ponging movements are

[3] An order n model is based on contexts of length n. With APC, the prediction agent is the same as the last element of the context, which is undefined in the case of a context of length 0.

[4] That is, an order 0 for MHC, or 1 for APC.

not removed; they show that ping-ponging is easy to predict, even with simple markovian models.

Results related to MHC are in accordance with [SKJH04b]. Since table 3.3 shows that ping-ponging has a major impact on the prediction results, it would be interesting to repeat the experiments presented in [SKJH04b] once ping-ponging has been filtered out.

The good prediction ratio is, overall, quite low. We can suppose that it would be higher for other kinds of networks where terminals are usually not switched off during their displacements (*e.g.* GSM), even if some of the terminals composing this dataset are WiFi phones ([KE02]). In this study, absolute accuracy performance is not our primary concern; we here emphasize the differences between APC and MHC schemes.

The decentralized MHC scheme works better than APC. This result was expected, as different persons have their specific behaviour: averaging the movement patterns of the people crossing the same AP only gives a rough estimate of the way they move. Surprisingly however, the accuracy difference between APC and MHC is small (from 55.4% to 59.8%); in practice, this means that getting decent prediction performance does not require to embed a prediction agent in each mobile: placing them in the fixed infrastructure can suffice. Section 3.5.2 gives hints on why the results of the two methods are so close.

3.5 Stressing the differences between APC and MHC

3.5.1 Prediction accuracy vs learning set cardinality

The markovian models' parameters learning process and the prediction process are *interleaved*: when a mobile is associated with an AP, this AP predicts where the terminal is going; as soon as the next AP is known, this piece of information is added to the learning set and allows it to improve the mobility model. The ratio of accurate predictions is thus a function of the elapsed learning time, and the way it evolves depends on the method used —APC or MHC.

Figure 3.4 (bold lines, left axis) draws the instantaneous prediction accuracy as a function of the learning sets cardinality. This plot shows that learning sets made of a few hundred elements lead to prediction ratios of more than 40%, regardless of whether prediction agents depend on APs or MHs. For bigger learning sets, both methods show different profiles. For APC, performance gets slowly better and stabilizes at 55% when the learning set is made of about 3000 samples. With MHC, the results increase much faster, reaching 60% when the learning set is composed of 1000 elements, and settling at about 73% with 2000 elements. This last percentage is astonishingly good and is thus studied more carefully below.

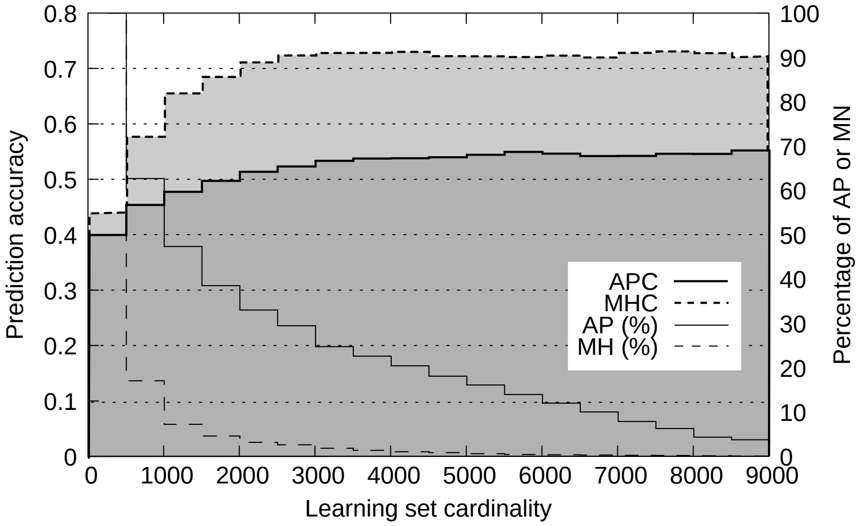


Figure 3.4: Prediction accuracy (strong lines, left axis) and proportion of prediction agents (thin lines, right axis) *vs* learning sets' cardinality using order 3 models.

On the same figure, thin lines (right axis) give the proportion of prediction agents which have a learning set cardinality higher than a given value. For example, for MHC, about 80% of the mobiles have a learning set composed of less than 500 elements. The curve associated with MHC decreases much faster than that of APC: nearly all the models have a learning set with a cardinality smaller than 3000. The APC curve has a very different shape: there are still some learning sets with cardinalities greater than 8000 elements. Only a small number of MHs are thus responsible for prediction ratios higher than 70%. A manual inspection of motion traces shows that ping-ponging between 3 APs or more *seems* to explain this anomaly. Fortunately, this situation is rare and should only marginally impact the average performance.

This figure allows finding the steady-state (*i.e.* on the long run) good prediction ratio reached by each method. If APC clearly settles at about 55%, the case of MHC needs to be considered more carefully. As mentioned above, performance of 70% or more are not realistic. We thus remove the 10% of best performing MHs and measure the performance of the remaining mobiles once they have reached a learning set cardinality greater than 500 elements; the good prediction ratio obtained is then 60.8%.

3.5.2 Next-AP distributions' entropy

Knowing the last APs encountered by a mobile terminal (or *context*) does not allow a perfect prediction of its next AP. This uncertainty can be formalized as a random distribution of next APs; this distribution is characterized by a given entropy. This entropy is commonly linked to the difficulty of predicting the motion of the mobile.

More formally, let $\mathcal{C}$ be the set of contexts, $N_a(c)$ the number of times context c has been observed by the prediction agent a, and $\mathcal{L}$ the set of APs. A prediction agent a is characterized by an entropy given by:

$$H_a = -\frac{1}{\sum_{c \in \mathcal{C}} N_a(c)} \sum_{c \in \mathcal{C}} N_a(c) \sum_{l \in \mathcal{L}} P_a(l|c) \log_2 P_a(l|c) \tag{3.3}$$

$P_a(l|c)$ is the probability estimated by a that l will be the next AP given the context c; it can be estimated using (3.2). Formula (3.3) thus computes the mean next-AP distribution entropy for each context, weighted by the context popularity.

Mean entropies of order 1, 2, and 3 models are given in table 3.4 for APC and MHC.

Method	**1**	**2**	**3**
APC	1.86 (0.95)	1.72 (1.18)	1.58 (0.49)
MHC	0.98 (0.50)	0.82 (0.66)	0.91 (0.37)

Table 3.4: Next-cell distribution entropies (in bits). Standard deviations are given between parentheses.

The two schemes exhibit strong differences. This runs counter to the results obtained in terms of prediction accuracy (see table 3.3) which showed a difference indeed, but as small as about 5%. From this experiment, we can conclude that MHC clearly predicts more precisely which APs might be next encountered by a MH, but this problem is different from the one studied in this here, which is only concerned with finding *the most probable* next AP. Thus, even if entropy estimation allows us to get a quantitative measurement of the mobiles' motion uncertainty given a model, and even if next-AP predictions accuracy is directly impacted by the model's precision, directly linking entropy to prediction accuracy gives a biased picture.

Figure 3.5 gives a clear view of how entropy is distributed among the APs. Each rectangle depicts an AP; its center's coordinates gives its entropy as computed by APC (x axis) and MHC (y axis). The x coordinate of the rectangle related to AP p is computed according to (3.3); for the y coordinate, we only take into account contexts ending with p. As expected, APC gives a entropy higher than MHC.

The rectangles' sizes are proportional to the number of "context, next-AP" couples that allowed us to estimate the entropy. To compute meaningful entropies, we do not allow an agent to report its entropy if it has been computed using less than 100 such couples. The dimension representing 25,000 couples is drawn near the graph's top-left corner. This explains the presence of flat rectangles: they match AP where most MHs passing by do not come back more than 100 times (notice that since they are crossed by a lot of different mobiles, APC often report an entropy higher than 3 bits). Popularity vary strongly between APs, a lot of

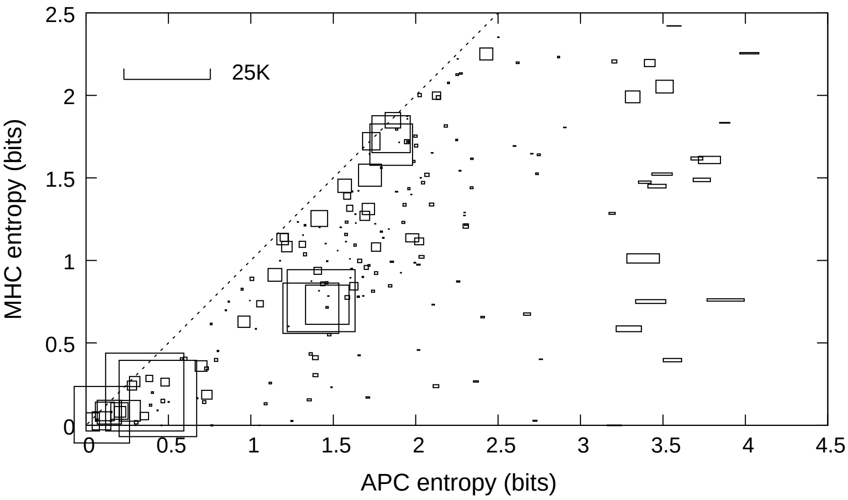


Figure 3.5: Next-AP distribution entropy for order 2 markovian models. The dotted line depicts the points where the next-cell distribution entropy is the same for both APC and MHC methods.

them being visited only from time to time, and a few being crossed more than 10,000 times.

3.5.3 Time division

It is commonly supposed (*e.g.* [CS98b]) that it is desirable to divide a learning set in homogeneous time slices: it seems for example sensible to expect different motion behaviours during the week-end and during the rest of the week, and it is thus reasonable to build different models for those periods of time.

Granularity	**APC**	**MHC**
No time division	*55.2%*	*59.8%*
Week/Week end	54.2%	58.4%
Morning/Afternoon	54.1%	58.2%
January 2003	53.8%	57.0%
Two hours periods	53.1%	53.4%
Days of week	52.0%	54.6%

Table 3.5: Prediction accuracy when different order 4 models are built for various time divisions. The results given are the (weighted) mean performance of the models built.

Using such a method would however bring two drawbacks: *(a)* the start of the model's learning curve could cause bad performance, and *(b)* short time periods could yield too small learning sets.

Table 3.5 gives the results obtained using various time divisions.

The results are surprising: in no case do the time slices improve the results. Two hypotheses can explain this fact: *(a)* the MHs' behaviours are the same during all the time periods (movements are not cyclic and can be described as a stationary process) or *(b)* the motion context already captures those differences.

3.6 Predicting the behaviour of new mobiles

A mobile terminal regularly connects with networks it has never seen before; it is usually the case of a user using a GSM phone on holidays. In these circumstances, the MHC method is of no use but APC could give satisfying results.

To simulate this situation, we divide MHs into two disjoint sets. The first is the *learning set* (LS, 5005 mobiles), the second the *test set* (TS, 1197 mobiles). The learning set allows us to build markovian models, as before. Those models are then used to predict the next AP of the mobiles in the test set.

Order	**LS**	**LS as TS**	**TS**
1	28.7%	28.8%	23.4%
2	47.6%	49.4%	39.0%
3	53.0%	58.9%	45.9%
4	55.2%	66.8%	48.5%
5	55.4%	71.5%	49.0%

Table 3.6: Prediction accuracy of APC using various test sets.

Table 3.6 compares the results obtained previously using APC (first column, they can also be found in table 3.3) with those obtained when the learning set is used as a test set (middle column), and those obtained using an independent test set (third column).

The very good result achieved when the learning and test sets are identical (middle column) reveals that simple models based on Markov chains can learn the characteristics of individual behaviours. As a corollary, using those models to predict the motion of mobiles that have never been seen before (last column) gives mitigated results that do not even reach 50%.

3.7 Conclusions and future works

Mobility prediction schemes can be divided into two main classes, here designated AP-centric (or centralized) and MH-centric (or decentralized). Quite surprisingly, they have never been directly quantitatively compared.

We have partially filled this gap using a study based on markovian models. The parameters of those models are fit *via* the analysis of a database containing the real motion traces of the mobile hosts of a campus WiFi network.

It allows us to draw a number of conclusions:

- Contrary to what one may have expected, the measured accuracy difference between APC and MHC is only a few percents (typically 55% vs 59%).
- In any case, the prediction accuracy is low (less than 60%); this is certainly a characteristic of WiFi users, and we expect other networks (*e.g.* GSM) to exhibit more predictable, regular motion patterns. This is not a real concern for this study as we are more interested in comparing APC and MHC rather than in absolute results.
- Next-AP prediction uncertainty can be estimated by an entropy measurement, but this only partially reflects prediction accuracy and, in this case, does not provide an accurate comparison of APC and MHC. One should thus refrain from linking entropy to prediction accuracy as this can introduce a bias.
- Quite surprisingly, we notice that building models specific to certain periods of time (week/week-end, morning/afternoon) does not bring any improvement.
- Using APC, next-AP prediction can be performed for mobiles that have never been connected to the network. The results are appreciably worse than when the MH are involved in the learning process, which shows that even simple markovian models can learn individual behaviours.

This comparison could be continued along several lines. For example, the relevance of an hybrid method combining APC and MHC schemes could be explored; the prediction of APC could, for example, be used when a MH reaches an AP it has never seen before. This situation barely arises in the dataset used here, hence such an experiment should be tried using traces extracted from a different network.

It would also be particularly desirable to reproduce the experiments presented here using other (more complex) models, different (more informative) pieces of information (*e.g.* measurements of the angle of arrival), and other kinds of wireless networks (*e.g.* GSM).

Part II

Performing Mobility Prediction

4

A Generic Mobility Prediction Method

. . . the question of whether Machines Can Think. . . is about as relevant as the question of whether Submarines Can Swim.

—EDSGER W. DIJKSTRA, *The threats to computing science* (1984)

In this chapter we present an original mobility prediction method. It has been designed to mitigate some shortcomings observed on other schemes, notably by means of genericity which is achieved on two levels:

- It is technology agnostic. Every mobility prediction algorithm depends on a source of information that allows it to monitor the mobiles motion; the algorithm presented here can handle various types of information sources without any modification.
- It does not rely on restrictive assumptions such as the knowledge of the AP coverage geometry.

Various simulations have been conducted to validate this scheme and to give an idea of its accuracy and robustness.

4.1 Introduction

We have already presented a number of mobility prediction methods in chapter 2. We have seen that they generaly rely on two assumptions:

- The mobile motion is monitored using some pieces of information. Those can be very precise (*e.g.* GPS positions), less informative (*e.g.* received signal strengths), or very fragmentary (*e.g.* previous AR(s), time of the day, time elapsed in the cell,. . . [HM99, CS98b]). *The type of information processed by the prediction scheme is assumed known in advance.*

- *The knowledge of the access points surroundings is assumed known.* For example, this knowledge can encompass the cell geometry (for cellular networks, see [LBC98]), or the road map of the cell coverage area (see section 2.1.5). In absence of such information, some methods rely on rough hypotheses, such as hexagonal cells organized in bee's nest; figure 4.1 illustrates how inappropriate those hypotheses can be certain environments.

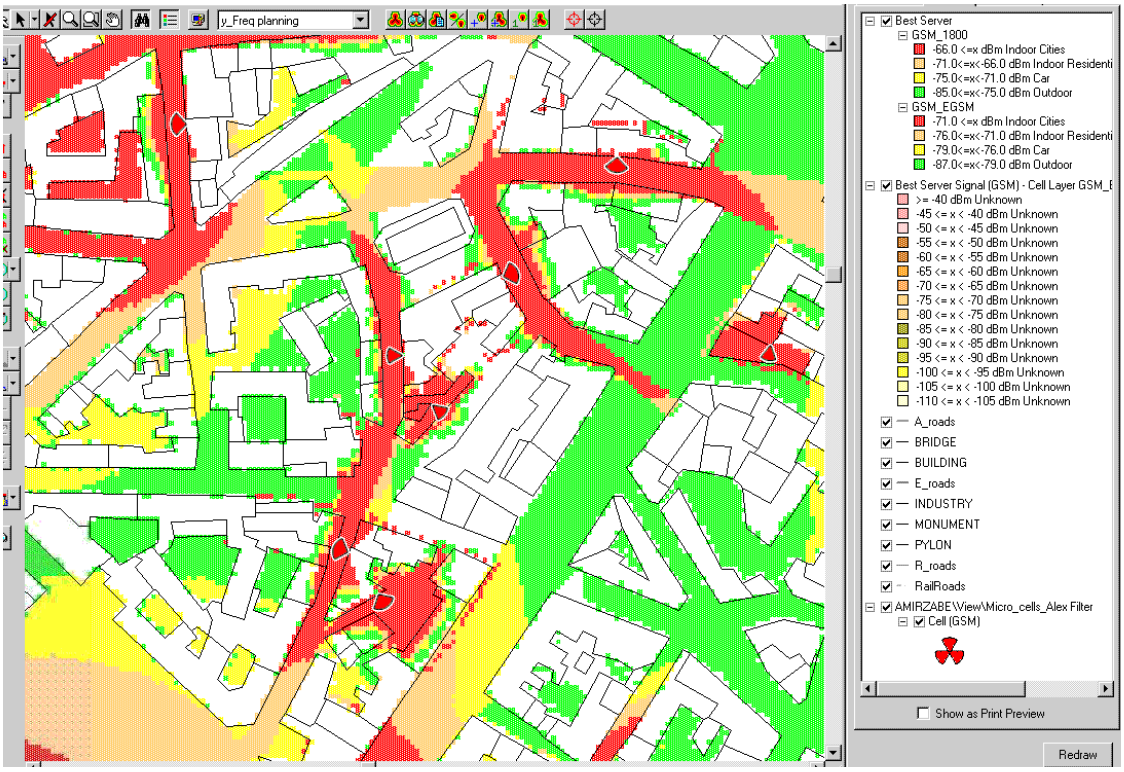

y_Freq planning
☑ Best Server
GSM_1800
🟥 -66.0 <=x dBm Indoor Cities
🟧 -71.0<=x<-66.0 dBm Indoor Residenti
🟨 -75.0<=x<-71.0 dBm Car
🟩 -85.0<=x<-75.0 dBm Outdoor
GSM_EGSM
🟥 -71.0 <=x dBm Indoor Cities
🟧 -76.0<=x<-71.0 dBm Indoor Residenti
🟨 -79.0<=x<-76.0 dBm Car
🟩 -87.0<=x<-79.0 dBm Outdoor
☑ Best Server Signal (GSM) - Cell Layer GSM_E
>= -40 dBm Unknown
-45 <= x < -40 dBm Unknown
-50 <= x < -45 dBm Unknown
-55 <= x < -50 dBm Unknown
-60 <= x < -55 dBm Unknown
-65 <= x < -60 dBm Unknown
-70 <= x < -65 dBm Unknown
-75 <= x < -70 dBm Unknown
-80 <= x < -75 dBm Unknown
-85 <= x < -80 dBm Unknown
-90 <= x < -85 dBm Unknown
-95 <= x < -90 dBm Unknown
-100 <= x < -95 dBm Unknown
-105 <= x < -100 dBm Unknown
-110 <= x < -105 dBm Unknown
☑ A_roads
☑ BRIDGE
☑ BUILDING
☑ E_roads
☑ INDUSTRY
☑ MONUMENT
☑ PYLON
☑ R_roads
☑ RailRoads
☑ AMIRZABE\View\Micro_cells_Alex Filter
☑ Cell (GSM)
□ Show as Print Preview
Redraw

Figure 4.1: Signal propagation simulation for GSM micro-cells at the centre of Brussels, Belgium (courtesy of BASE®). Notice that antennas are not distributed in bee's nest, nor is the propagation uniform (streets behave as wave guides in urban environments).

We here propose a *generic* mobility prediction method that removes those assumptions. We aim for a method that can easily be adapted to various kinds of networks, *i.e.* we seek technology independence.

In order to get genericity, we rely on a simple artificial intelligence technique that allows us to directly link the mobile motion observation to the handover mechanism.

To be as generic as possible, the movement patterns learning process presented here tries and accomodate any type of information emitted by MHs, and gives a simple statistical result when there is none. It is based on a simple AI technique and gives to mobiles a simple role.

4.2 Movement patterns and Markov processes

Let's consider somebody walking down a street using a mobile device connected to a wireless IP network. This device communicates with a given *Access Point* (AP) and can regularly measure some information directly related to the path it follows (*e.g.* given by the layer 2 protocol or a separate apparatus such as a GPS).

Reporting these measurements at discrete time steps $t = 1, 2, 3, \dots$, we get a sequence of observations (respectively $O_1, O_2, O_3, \dots$), each of them being a vector of discrete or continuous values.

Since we are interested in the paths followed by mobiles, let's divide the zone near an AP in small areas (or *states*) Q_i and try to match those states with sequences' observations. Let q_t be the state matching the tth observation (see figure 4.2).

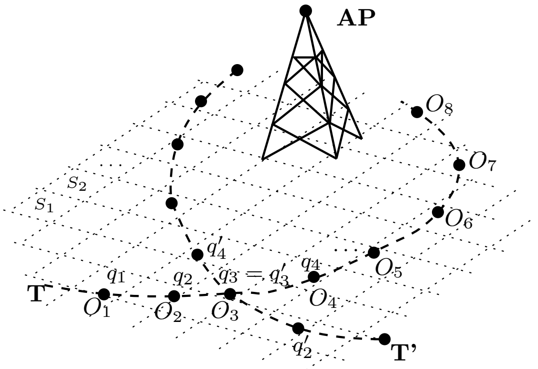


Figure 4.2: Two trajectories near an AP and the associated set of observations and states.

If mobiles movements were memoryless, *i.e.* if:

$$P(q_t = Q_i \mid q_{t-1} = Q_j, q_{t-2} = Q_k, \dots) = P(q_t = Q_i \mid q_{t-1} = Q_j) \quad (\forall t) \qquad (4.1)$$

then it would be a first order Markov chain and the model shown hereafter would be particularly well suited. Unfortunately, the simple example at figure 4.2 shows that (4.1) does not hold (one can't distinguish between trajectories T and T' knowing q_3 or q_3' only, but can if it's given q_2 or q_2'). Despite that, simulations under different conditions show that it gives good results if it's used as explained later on (see section 4.4.1).

Considering a walker without any a priori information linked to an AP for a given time, we can get an observation sequence S. S is stochastic for at least two reasons:

a. each observation can be corrupted by noise or some other kind of measurement error.

b. nobody walks randomly, but usually several paths could be followed in an AP's zone of influence.

Because of a., the state-to-observation matching can only be probabilistic, so we'll define a *observation's probability function* of a given state. Because of b., guessing q_t knowing q_{t-1} can't be done exactly, so we'll define a *state transition probability matrix*.

In practice, the different states q_i a mobile is passing by are hidden, but can be deduced from observation sequences. This kind of (doubly stochastic) process can be modelled using *Hidden Markov Models* (HMMs).

4.3 Markov processes and HMMs

4.3.1 Overview of HMMs

A HMM ([RJ86, Rab89]) is a model of observation sequences $S = O_1, O_2, O_3, \dots$ as explained in section 4.2. It is characterized by:

- *N states*: $Q_1, Q_2, \dots, Q_N$. The state at time t is denoted q_t;
- *a set of observation's probability distibution functions*: $B = \{b_i(O)\}$.

$$b_i(O) = P(O \text{ at } t \mid q_t = Q_i) \qquad \forall i = 1, 2, \dots, N \tag{4.2}$$

- *a state transition probability matrix*: $A = \{a_{ij}\}$.

$$a_{ij} = P(q_t = Q_j \mid q_{t-1} = Q_i) \qquad \forall i, j = 1, 2, \dots, N \tag{4.3}$$

The following properties hold:

$$0 \leq a_{ij} \leq 1 \qquad \forall i, j = 1, 2, \dots, N \tag{4.4}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \forall i = 1, 2, \dots, N \tag{4.5}$$

- *the probability that Q_i is the initial state of a sequence*:

$$\pi_i = P(q_1 = Q_i) \qquad \forall i = 1, 2, \dots, N \tag{4.6}$$

Let π be the set of these probabilities: $\pi = \{\pi_i\}$.

A HMM model is usually denoted $\lambda = (A, B, \pi)$ which reminds of the parameters involved.

The probability of observing a sequence S and a sequence of states $Q = q_1, q_2, q_3, \ldots q_T$ given a model λ is simply the probability that q_1 is the initial state, times the probability of observing O_1 at q_1 (hereafter denoted $\pi_{q_1} = \pi_i$ with i such that $q_1 = Q_i$), times the probability of going from q_1 to q_2,…:

$$P(S \mid Q, \lambda)\ P(Q \mid \lambda) = \pi_{q_1} b(O_1) a_{q_1 q_2} b(O_2) a_{q_2 q_3} \ldots a_{q_{T-1} q_T} b(O_T) \tag{4.7}$$

To get the probability of S given λ, we simply sum the probability of all the possible state sequences:

$$P(S \mid \lambda) = \sum_{\text{all } Q} P(S \mid Q, \lambda)\ P(Q \mid \lambda) \tag{4.8}$$

Another problem of great interest is: how can one find the most probable state sequence matching a given observation sequence S:

$$Q^* = \underset{Q}{\operatorname{argmax}}\ P(S, Q \mid \lambda) \tag{4.9}$$

The calculation of (4.8) and (4.9) are well known problems that have been resolved efficiently ([Rab89]).

4.3.2 HMM training

The last unresolved question is: how can one adjust the model parameters of $\lambda = (A, B, \pi)$ to best suit a set of observation sequences? It depends on what *best suits* means in this context and it usually has no exact analytical solution. An approximated estimate can however be found using iterative algorithms that seek for locally optimal solutions.

An often used optimization criterion is the *maximum likelihood* (optimizing $P(S \mid \lambda)$), yielding to the Baum-Welch algorithm. Another criterion is the *state optimized likelihood* (optimizing $P(S, Q^* \mid \lambda)$, where Q^* is given by (4.9)), which lead to the k-means algorithm. This last algorithm has been used in this work; it is briefly described in figure 4.3.

This work has been an opportunity to program the classical algorithms related to HMMs in the open source *Jahmm* library (appendix B gives more details). This library has been used in various research activities ([ZBRC06, LC05, LH05, CKP$^+$06]…) and as an educational tool for several machine learning courses.

4.4 HMMs and path prediction

4.4.1 Overview

In mobile IP networks, access routers manage mobile nodes arrivals and departures. Using this method, the movement pattern learning takes place in each AR, so that the MN's role is kept minimum.

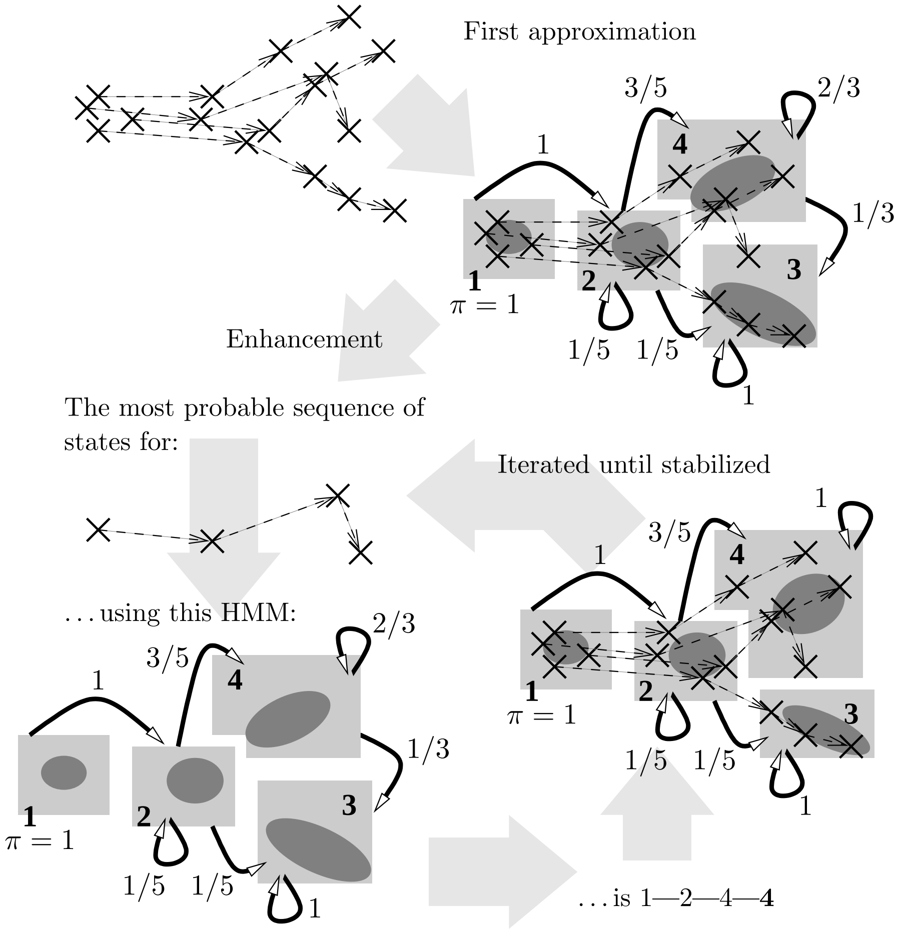


Figure 4.3: The k-means training algorithm ([JR90]). Let's imagine four cars that regularly deliver their coordinates; they all come from the west, three of them go to the north-east, on to the south-east (see the top-left diagram). One can get a first approximation of the resulting HMM by clustering the observations (top-right diagram, rectangles in light grey); each of those groups matches a state of the HMM (here, it has been chosen to use four states). We deduce: *(a)* the weights of the state transitions (directly extracted from the state sequences), and *(b)* the probability density functions, estimated from the observations matching the state (here depicted by an error ellipse of a normal distribution). After that, one can compute, for each observation sequence, the associated most probable state sequence (equation (4.9), only one sequence is depicted on the diagram); one can them compare, for each observation, its most probable state and the state that had been associated at the previous step. If they don't match, the HMM is modified and re-estimated. This process is repeated until no more modification takes place. The algorithm convergence is guaranteed for discrete variables; continuous variables require additional hypothesis on the b_i functions ([Rab89, JR90]).

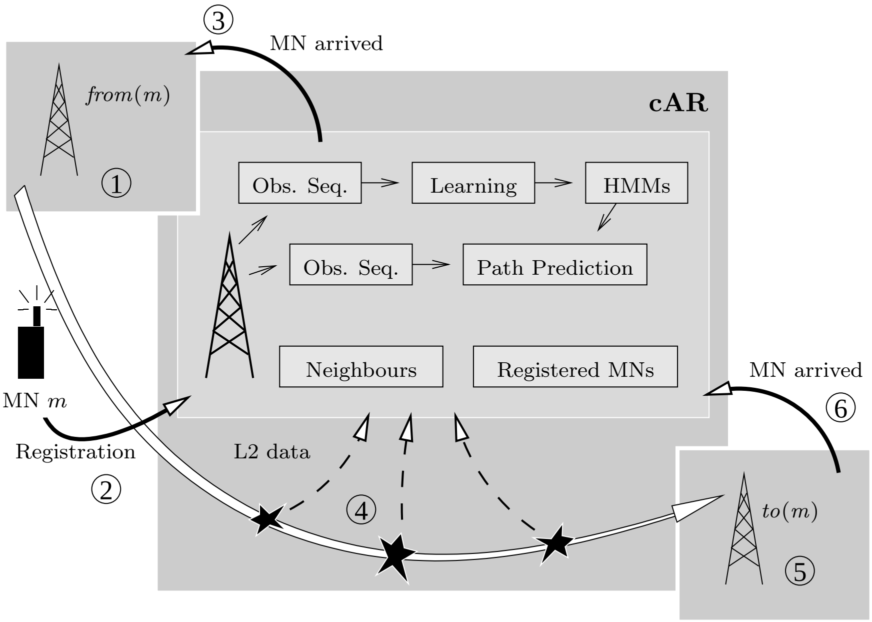


Figure 4.4: Overview of the proposed mobility prediction architecture. Once cAR has seen enough MN's passing by in order to learn their typical behaviour, the path prediction procedure happens as follow: (1) A MN m leaves the AR *from*(m) and goes towards cAR. — (2) It registers with cAR. — (3) cAR informs *from*(m) about m's arrival. It adds *from*(m) to its neighbours list (if it has not been done before, which is unlikely since we suppose that the learning process is terminated). — (4) From time to time, m emits observations about its journey; this lets cAR build a sequence and check its probability using the HMMs. When it appears that one HMM is clearly more probable than the others, the matching neighbour can be informed. — (5) The same procedure takes place when m leaves cAR and reaches *to*(m). — (6) *to*(m) informs cAR about m's arrival (just as cAR did).

In this context, the path prediction's purpose is twofold:

- learning the typical MN's movement patterns so as to guess their future AR;
- sending to each neighbouring AR statistical information related to MNs that are likely to turn towards it (possibly together with related indication, as the needed QoS).

We say AR_j is *in the neighbourhood of* AR_i if mobile nodes regularly travel from AR_i to AR_j; therefore, the neighbouring relation is dynamic and must be constantly reevaluated. To let an AR know its neighbours, we suppose that every MN remembers the last AR he was registered with; this way, a neighbour can give an AR notice of a mobile's arrival.

This neighbouring relation is a key point here, and it is determined by the way the mobiles travel between the ARs. Let R be the set of MNs registered with the current AR *cAR* (*i.e.* the one we are interested in). We denote *from*(m) the AR a MN m visited just before the current one; similarly, *to*(m) is the next AR m will register with. The current AR's set of neighbours is denoted N; thus, $\forall m \in R : \mathit{from}(m) \in N$ and $\mathit{to}(m) \in N$.

It could be possible to catch the behaviours of all the MNs an AR is involved with using only one HMM, but we don't think this is the best way to organize the learning process. In order to get observation sequences as close as possible to (4.1), one shouldn't mix unsimilar sequences. It is thus proposed to create a different model for sequences built by mobiles which are not going to or coming from the same AR; so, the number of HMMs grows as the number of neighbours squared. Let $\lambda_{i,j}$ be the HMM associated with the mobiles going from AR_i to AR_j. We will see that the number of sequences used to build $\lambda_{i,j}$ will take some importance; let $p_{i,j}$ be that number and p_i their sum $(\sum_{j\in N} p_{i,j})$.

Figure 4.4 gives an overview of the different actions involved when a MN m is passing by the access routers *from*(m), cAR and finally *to*(m). On its travel, it can send L2 information (*e.g.* piggybacked with other kinds of data) which are received and stored by cAR. When registering with *to*(m), the MN sends the address of the AR he just left (cAR) together with its Home Address; then, *to*(m) informs cAR about the MN's arrival, so that it can:

- add *to*(m) to its neighbours list. This list is regularly cleaned up so as to remove unrefreshed entries;
- match the MN's home address with the addresses of recently-gone MNs and find the associated (L2) observation sequence;
- use it to build $\lambda_{i,j}$. It cannot be done before enough sequences have been collected.

Once the learning process is stabilized, the path prediction itself takes place. A MN m generates a sequence $S = O_1, O_2, O_3, \ldots$; let S_i be S truncated so that only the first i observations remain. Each time an new observation is emitted, the AR can compute

$$P_j = P(S_i \mid \lambda_{from(m),j}) \qquad \forall j \in N \tag{4.10}$$

using (4.8). The ratio between those probabilities weighted by $p_{from(m),j}/p_{from(m)}$ tells us towards which neighbour m is likely to go, and the certainty of doing so.

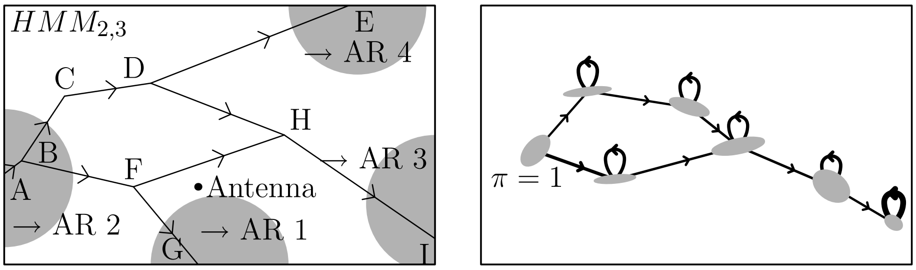


Figure 4.5: A simulation of path learning. The left picture shows a map of one-way roads. The cars that are coming from A and find their way to E, G and I at random. As they are passing by, they generate sequences of observations; those sequences can be learned by a HMM, as shown in the right figure. Each state is drawn using its gaussian observation probability function's covariance ellipse (enlarged 1.5 times for clarity).

Let's consider the issues that could prevent the method from scaling. The only messages needed to apply the method are shown in figure 4.4; the amount of the data sent during the registration process is small and the various data sent during the MN journey can be piggybacked (for example, the Mobile IPv4 registration process requires 116 bytes).

The estimation of the parameters of the HMMs given in the next section takes less than 1 second per model[1]; furthermore, the learning process can be done *off-line*, on a representative sample of the learning set. Computing the probability of a sequence (given a model) can be done in a few milliseconds. The complexity of the algorithm is N^2T, N being the number of model's states and T the length of the sequence.

4.4.2 First simulations

Figure 4.5 shows the results given by a simulation of the above procedure. Cars equipped with a GPS are passing by an AR; thus, the observations are 2-dimensional (x, y) vectors giving the position of the car. To be as generic as possible, we suppose that all the observation probabilities can be modeled using a gaussian probability

[1] Computation done with 400 sequences using a Java prototype on a Pentium 1.2GHz.

function; thus, results could be made even better if probability functions were built more carefully, at the expense of generality.

The left picture shows the topology of some one-way roads. The vehicules come from A and travel as indicated by the arrows; at each crossing, the cars choose their way randomly. The speed of each car is a random constant (linearly distributed between a minimum and a maximum value) and observations are emitted regularly. We can be given an insight into the computed HMM using a graphical representation (see the right picture): the transition probabilities are depicted by lines of different widths, and the probability distribution of each state by a covariance ellipse[2].

As one can see in figure 4.6, for the example given, the HMM easily discriminates the different sequences and the next-neighbour prediction should never fail.

Now, an important point is: how can we replace the GPS data with a different one, for example the mobile-antenna distance (which could be derived from the received antenna power)? In fact, nearly no modifications is needed at all: the HMMs just learn 1-dimensional vectors instead of 2-dimensional ones. Figure 4.7 shows the results with this new configuration; they should be compared with those of figure 4.6.

Simulations of this technique have been realized (using mobiles-antenna distances) with several maps of different complexity. With simple maps, such as the one presented at figure 4.5, the prediction gives nearly perfect results (between 98 and 100%). A more complex map made of 40 intricated roads and 7 potential neighbouring ARs shows the importance of the sequences length and HMM size: 5 states HMMs modeling sequences of 5 observations on average guess the correct next neighbour 67% of the time. This number is as high as 82% when using 10 states HMMs with sequences of 10 observations on average. Interestingly, when counting as a good guess the actual next AR being one of the first two most probable next neighbours, those figures grow to 88% and 94% respectively. Mixing short sequences with large HMMs (or the other way round, long sequences with small HMMs) gives intermediate results. The influence of noise can be reduced using longer sequences, up to a certain level where the sequences become indistinguishable.

4.5 Simulations

4.5.1 Principle

To perform mobility predictions in a more realistic settings, what precedes has to be a little bit refined.

[2] The *covariance ellipse* (or *error ellipse*) of a bivariate gaussian probability function is an ellipse of constant probability density, centered at each variable's mean, and such that the probability of an observation to be in this ellipse is ≈ 0.39.

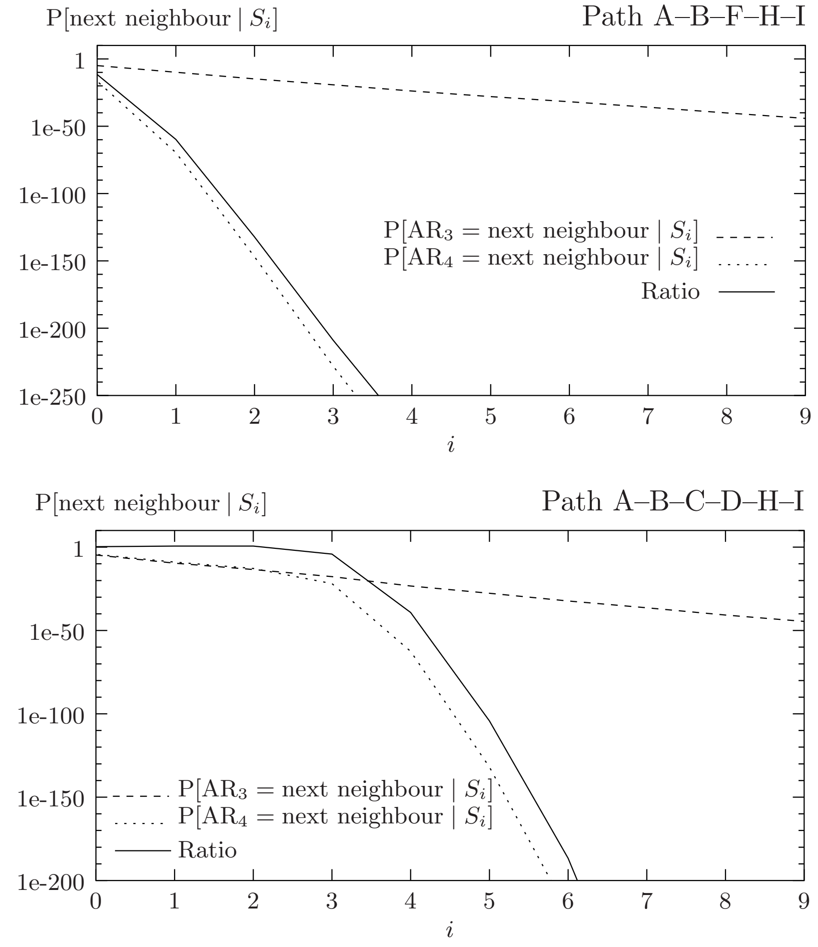
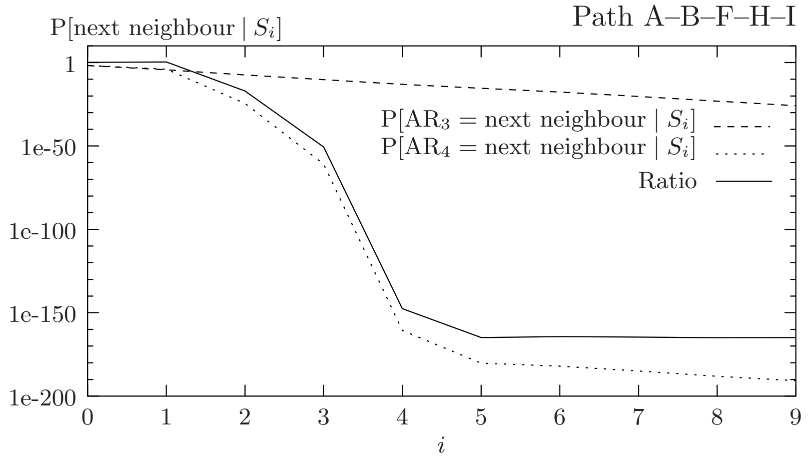


Figure 4.6: Prediction (using GPS coordinates). Those graphs show the evolution of $P(S_i \mid \lambda)$ with i (*i.e.* how the next-neighbour predition evolves as the time passes) for a mobile that goes from A to I (see figure 4.5). The two dashed curves correspond to the probability that the sequence belongs to each of the two given HMM; the plain one is their ratio (and should be smaller than 1 if the prediction is right). The top graph shows that when a mobile chooses to reach I *via* B, F and H, the prediction is easy because this path is very different from the one that reaches E (A-B-C-D-E). On the contrary, the bottom graph shows that while the mobile travels towards E, the probabilities ratio stays close to 1 during the first 3 observations.



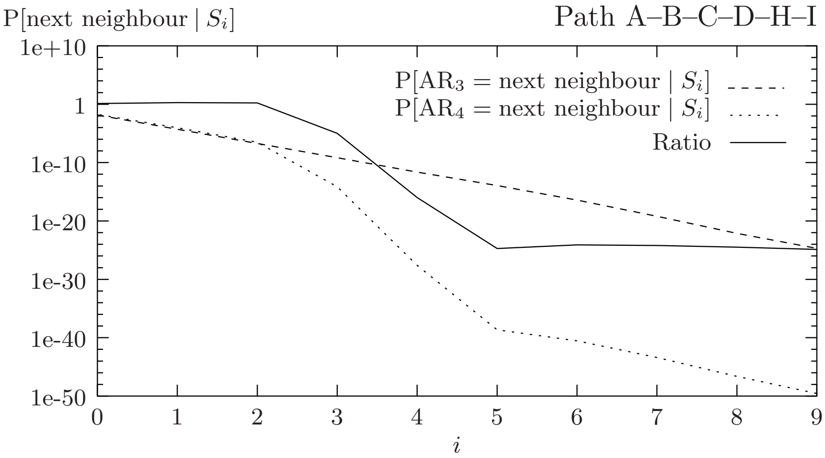


Figure 4.7: Prediction (using antenna distance). Those graphs are very similar to those of figure 4.6. Here, the observations are the distances between the mobile and the antenna it is linked to (this antenna is situated near the middle of the road map; see figure 4.5). The prediction process behaves much like with a GPS, but is less precise.

Let $S_{i,j}$ be the set of observation sequences matching the mobiles moving from AR_i to AR_j. Let $n_{i,j}$ be the number of sequence of $S_{i,j}$.

As explained above, the training of $\lambda_{i,j}$ is performed thanks to $S_{i,j}$ (see section 4.3.2). The learning time period should be short enough to be able to adapt to the varying motion behaviour along the day; one must take care of keeping a learning set large enough so that it can be used to extract the typical motion habits.

Once the prediction process takes place, the most probable next neighbour must be estimated using (4.10) which must be slightly modified to cope with the fact that some neighbours might be more popular than others.

$$P_j = \frac{n_{from(m),j}}{\sum_{i=1}^{n} n_{from(m),i}} \; P(S_i \mid \lambda_{from(m),j}) \qquad \forall j \in N \tag{4.11}$$

4.5.2 Observations without noise

One of the main factors that can impact the next neighbour prediction accuracy is the complexity of the road map in the area considered. This is easily understandable: increasing the number of potential paths leading to a next neighbour leads to HMMs which must be able to recognize thoses trajectories; the HMMs are thus more complex and the probability of seeing two different paths (leading to different neighbours) producing similar observation sequences is increased.

We have build several maps of increasing complexities. On each of them, the mobiles move at a constant (yet random) speed along the roads, following the arrows, and choose their way randomly each time they reach a crossroad.

Another factor likely to impact the method is the mean length of the generated sequences (*i.e.* their number of elements). Intuitively, longer sequences give a better view of the underlying trajectory and, thus, help the prediction process. In the following simulations, observations are generated every time step. The duration of a time step thus allows to change the mean sequence length.

The value of an important parameter has to be chosen: the HMMs number of states. The higher the number of states, the more precise the model. The training time of the HMMs of course increases with the number of states, but the processing time requirements are not a big concern in this context: HMMs are only re-evaluated from time to time, when the MHs motion behaviour might have changed.

Figure 4.8 gives the results of the simulations undertaken.

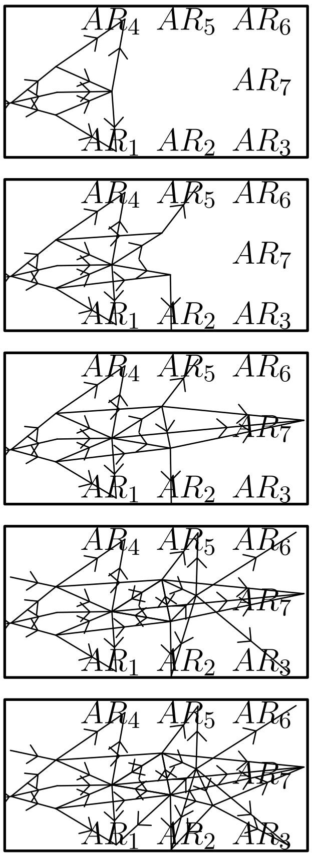
The simulations training conditions are as follows: the mobiles move according to the maps depicted figure 4.8. During their journey, they emit observations at regular time intervals. Once they reach the map border, the AR towards which they are headed is determined and the observation sequence is collected. This sequence is used by the training/prediction process as explained above.

In figure 4.8, the “training” column gives the results obtained thanks to a training based on the distance between the MH and the antenna they are linked to. The training has been done with 5 and 10 states HMMs, and with sequences made of 5 or 10 observations.

The “statistical” column gives the performances obtained if the MHs cannot emit any observation. In this case, the second term of (4.11) is simply equal to 1, and the prediction is purely statistical. This method might appear rather radical at first, it is however the only one that can be used in the presence of “blind” terminals (such as a laptop without any wireless interface), an important particular case. One should notice that those statistics are useful to put anticipated shared resources in place (see [CS98b, HM99]), and that they are directly accessible in the framework presented here.

Formula 4.8 allows build a list that ranks the potential next access routers according to their probability. The numbers included in the columns titled “1st” are

Map Learning Statistical



Size	5 states		10 states	
	1st	2nd	1st	2nd
5	100%	100%	99%	100%
10	100%	100%	100%	100%

1st	2nd
51%	100%

Size	5 states		10 states	
	1st	2nd	1st	2nd
5	89%	98%	97%	99%
10	94%	100%	95%	100%

1st	2nd
26%	51%

Size	5 states		10 states	
	1st	2nd	1st	2nd
5	91%	98%	91%	100%
10	91%	98%	93%	100%

1st	2nd
33%	55%

Size	5 states		10 states	
	1st	2nd	1st	2nd
5	67%	88%	78%	91%
10	79%	93%	81%	95%

1st	2nd
30%	59%

Size	5 states		10 states	
	1st	2nd	1st	2nd
5	67%	88%	72%	87%
10	78%	95%	82%	94%

1st	2nd
30%	57%

Figure 4.8: Simulations with noiseless observations. The left column depicts the maps used during the simulations; complex maps are supersets of simpler ones. The simulation results are given in the other two columns. The middle one is related to simulations based on the distance between the mobiles and the AP; the right one is a purely statistical prediction (no observations are emitted). The columns labeled "1st" give the prediction accuracy, those labeled with "2nd" consider a prediction as accurate if the next AR is the most likely or the second most likely prediction.

the percentages of accurate predictions — *i.e.* such that $to(m)$ is the first element of this AR list. The columns titled "2nd" give the same piece of information, but $to(m)$ can this time be the first or second element.

The first map is quite simple but represents an important case since it is common to see large areas with a low population density. Those are likely to have few roads and to be characterized by simple mobile trajectories. In those simple conditions, the prediction scheme behaves almost perfectly. The results of the right column is of course a consequence of the map symmetry.

The next two maps increase the number of neighbouring ARs. The results continue to be satisfactory: one can observe the efficiency of simple Markov models associated with short observation sequences. From a statistical point of view, AR_7 is the most probable next AR in the third map; this helps the prediction agent despite the higher number of potential destinations. The motion model chosen here is completely random, even if in reality some roads are more popular than others; it can thus be seen as pessimistic.

The last two maps correspond to quite difficult problems: the MHs can enter the map using two different entry points, several roads and destinations have been added, and the road graph is now cyclic. Simple models (few states, short sequences) show their limits, while more complex methods continue to behave quite satisfactory. One can see that long sequences are now mandatory in order to discriminate between complex trajectories.

4.5.3 The impact of noise

Figure 4.8's most complex map has been used to analyze the impact of noise on HMMs performance.

This noise is added to observations in an aggressive manner: it is proportional to the signal value according to the following formula:

$$x_b = (1 + G_{0,1} f)x$$

where $G_{0,1}$ is a random normal distribution with a mean $\mu = 0$ and an unitary variance $\sigma = 1$.

Notice that we are not interested to modeling accurately the noise typically encountered on wireless channels; we just express it reasonably yet pessimistically.

Simulations have been done with various values of f; the results are given figure 4.9.

4.5.4 Comments on the results

As expected, the results are always better with longer sequences or more HMM models using more states.

$f = 0\%$

Size	5 states		10 states	
	1st	2nd	1st	2nd
5	67%	88%	72%	87%
10	78%	95%	82%	94%

$f = 5\%$

Size	5 states		10 states	
	1st	2nd	1st	2nd
5	64%	88%	69%	86%
10	71%	92%	71%	93%

$f = 10\%$

Size	5 states		10 states	
	1st	2nd	1st	2nd
5	58%	83%	61%	81%
10	59%	87%	64%	87%

$f = 15\%$

Size	5 states		10 states	
	1st	2nd	1st	2nd
5	50%	75%	54%	75%
10	51%	81%	53%	83%

Figure 4.9: Simulations with noisy observations.

In every case, the method performs way better than a simple, statistical approach.

The method can give satisfactory results even when used with short sequences: in the simulations presented here, short sequences lead, in the worse case, to a difference of 15%.

If the length of the observation sequences cannot be changed, increasing the HMMs number of states is advisable. Symmetrically, using long sequences with simple HMMs is useful.

Adding noise to the observations decreases the method accuracy: with complex maps and the type of observations studied here, the different trajectories become indistinguishable. In this case, it is mandatory to use the most two likely next ARs instead of only one.

4.6 Extensions

One could argue that more information about mobiles displacements are needed than just the different next-neighbour probabilities. For example, a potential problem is to estimate when a handover will occur so as to prepare it ahead of schedule, but not too much.

Another problem is how to handle mobiles with no means of generating any observation at all. In this case, we must resign to do a statistical estimation of the mobiles movements (an example of how useful this information is can be found in [CS98b]).

Those problems are studied in the next sections.

4.6.1 Learning more

This section describes how to take advantage of the MNs' path learning to learn other kinds of information. For example, suppose one wants to match an observations sequence S (of length i) with the expected amount of time before the next handover. The solution is straightforward: apply the same procedure as we've just seen, and, before they enter the HMM learning process, add a dimension (the time we are interested in) to each observation vector. Thus, if we consider that the observations are emitted at a constant rate, this conversion can be written:

$$S = \begin{pmatrix} o_1^1 \\ \vdots \\ o_d^1 \end{pmatrix} \begin{pmatrix} o_1^2 \\ \vdots \\ o_d^2 \end{pmatrix} \cdots \begin{pmatrix} o_1^i \\ \vdots \\ o_d^i \end{pmatrix} \rightarrow$$

$$S' = \begin{pmatrix} o_1^1 \\ \vdots \\ o_d^1 \\ t \end{pmatrix} \begin{pmatrix} o_1^2 \\ \vdots \\ o_d^2 \\ \frac{i-2}{i-1}t \end{pmatrix} \cdots \begin{pmatrix} o_1^j \\ \vdots \\ o_d^j \\ \frac{i-j}{i-1}t \end{pmatrix} \cdots \begin{pmatrix} o_1^i \\ \vdots \\ o_d^i \\ 0 \end{pmatrix} \tag{4.12}$$

... where t is the time taken to generate S (from registration to handover). After that, one can use (4.9) to guess the most probable state a given sequence ends with, and see the associated time distribution.

Figure 4.10 shows the results given by this method. Tests have been produced with two HMM models: $\lambda_{2,3}$ with 7 states (as in fig. 4.5) and with 15 states. The t axis reflects the exact time-to-go when the observation is emitted; μ_t and σ_t are respectively its approximated value and standard deviation computed with the HMMs. As one could expect, increasing the model's complexity leads to more accurate time predictions and smaller deviations.

4.6.2 Statistical prediction

If we aim at building a prediction mechanism as generic as possible, we must look at the important case of mobiles unable to generate any kind of hint about their movements.

First of all, considering a MN m, notice that $\mathrm{P}[\mathit{to}(m) = \mathrm{AR}_j \mid \mathit{from}(m) = \mathrm{AR}_i] = p_{i,j}/p_i$ holds and that the sum of those probabilities $\forall m \in R$ gives the expected proportion that will reach AR_j. Clearly, we would get a more precise result if we could take into account for how much time the MNs is registered.

We can consider the situation depicted here as a particular case of the section 4.6.1 above, with sequences S_i containing an observation vector of dimension 0. As before, we add the time elapsed between the mobile registration and its handover to this null observation; simple single-state HMMs are sufficient to learn those degenerated sequences.

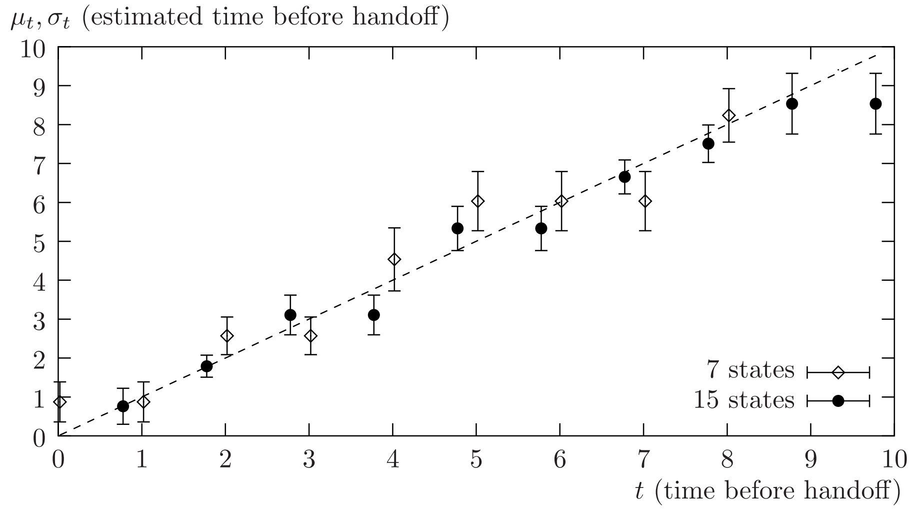


Figure 4.10: Departure time prediction. Two sequences have been generated by two MNs travelling on the same road, at the same speed. The graph shows the time-to-go predicted by two HMMs with different number of states. The computed standard deviations are given by the bars above and underneath the dots.

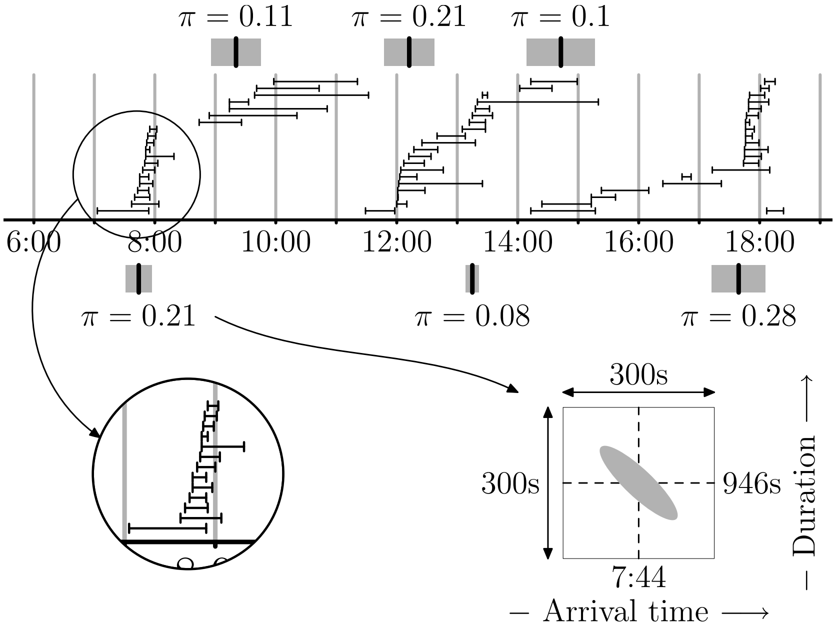


Figure 4.11: Departure prediction. Each segment above the timeline shows when a MN arrived and how long he stayed. When feeding a HMM learning algorithm with this information, one gets a set of states depicted using grey rectangles crossed by a line; the line (resp. grey zone) represents the mean arrival hour (resp. arrival hour's standard deviation) of its observation probability function (for a mean registration-to-handover delay). The left graph shows the covariance ellipse of the most probable state for observations emitted at about 8:00.

Once the learning process has taken place, one can use the probability function $b_{from(m),j}$ associated with $\lambda_{from(m),j}$'s single state to compute the probability a given MN m has to do a handover to reach AR$_j$ in a timeslot of T time units if it arrived t_m time units ago (denoted $p_j^{m,T}$; the sum of these values yields the expected number of mobiles that will travel to AR$_j$ in the next timeslot of T time units (denoted n_j).

$$\begin{aligned} p_j^{m,T} &= \frac{p_{from(m),j}}{p_{from(m)}} \frac{\int_{t_m}^{t_m+T} b_{from(m),j}(x)\,\mathrm{d}x}{\int_{t_m}^{+\infty} b_{from(m),j}(x)\,\mathrm{d}x} \\ n_j &= \sum_{\forall m \in R} p_j^{m,T} \end{aligned}$$

If one needs to predict the bandwith taken by MNs going to a particular neighbour, then using 2-dimensional vectors (holding the registration-to-handover delay and the bandwith used) is well suited. The covariance of those vectors' gaussian model could point out, for example, that fast MNs usually use more bandwidth (*e.g.* because high speed trains give wireless multimedia services).

In a wired environment where MNs are, for example, laptops liable to leave the network at any time, HMMs could be used to predict departure hours using the learned visiting habits. With that aim, the sequences can be made of vectors like this:

$$\begin{pmatrix} \text{registration-to-handover delay} \\ \text{registration time of day} \end{pmatrix}$$

Figure 4.11 shows the results given by this method; it depicts what might be the visits to a students' Internet room in a campus. One can see that states are regularly distributed over the day, allowing us to discover some typical movement behaviour. The left-hand “8 o'clock probability function” illustrates this point: it appears that, just before 8 (beginning of the first course), several short visits are made (*e.g.* to get emails) and that the later they begin, the shorter they are (so as not to be late).

4.7 Conclusions and future work

In this chapter, the choice has been made not to rely on the MNs to know their typical path, but rather to let the ARs do the learning process. This choice is meant to save MNs' computation time and/or bandwith on wireless channels.

This approach yields to a fairly generic method and allows one to discard some hypotheses (*e.g.* the knowledge of the cell geometry).

An overview of how to apply a well-known, simple learning method (*i.e.* Hidden Markov Models) to the path prediction problem in mobile networks has been given.

It has been shown that HMMs give good results, and can be extended to cope with typical problems involved in this context. In particular, the special case of MNs in wired networks has been studied.

Simulations have been done using motion maps of various complexities; we have shown how the method can be impacted by noise.

In the future, tests with real-life data should be undertaken, so as to see the differences with the simulations proposed here.

In the presence of mobiles regularly reporting their location, the resulting HMMs could be used as the basis of a *mobility model*, *i.e.* a description of terminals motion. Unlike other mobility models ([HGPC99, Bet01, CBD02]) the mobiles' motion behaviour would not be set *a priori*, but could be learned using real mobility traces.

5 Entropy-Based Knowledge Spreading

A year spent in artificial intelligence
is enough to make one believe in God.
— ALAN J. PERLIS, *Epigrams on Programming* (1982)

We have already mentioned that the foundation of most mobility prediction schemes is the indirect observation of the mobile hosts motion. This observation is done by means of various sources of information, such as signal strength measurements. We call those observed pieces of information “clues”.

Some of those information sources are likely to be more informative than others. Since mobile terminals are often memory and bandwidth limited, it is desirable to remove the clues that convey a limited amount of information to the prediction process.

This is the aim of the scheme proposed in this chapter. We let the system discover the most relevant clues for performing mobility prediction, and find how to propagate them sensibly in the network. Using a simulation, we show that it helps improve the prediction accuracy on small devices.

5.1 Introduction

The kind of information used to perform this monitoring is varied: GPS coordinates, motion speed, access router identification, signal strength,. . . Obviously the information about imminent handovers brought by each of those information types is not the same; some are more relevant — or more precise — than others.

Since mobile networks often involve small, limited devices, it is important to design mobility prediction schemes that are as accurate as possible without wasting valuable memory. One way to achieve this goal is to carefully select the most interesting pieces of information and to discard the others.

In chapters 2 and 3 we have stressed that mobility prediction techniques usually use one of the two following schemes:

- Either mobiles store their most frequent paths locally (at the expense of the memory consumed) or they get them from a "home repository" (increasing delays and consuming bandwidth [PGM04, LBC98]);
- Or prediction is done locally, usually in each access router, using the typical behaviour of mobiles encountered in the past ([SK04]).

Both methods have pros and cons. Keeping pieces of information — such as the frequent paths of mobiles — for long periods of time can be profitable if they help guess mobile movements later on.

Ideally, each mobile would know where each piece of information could be exploited and only keep those having a high probability of being used. This leads to a hybrid method which combines the advantages of both schemes. This hybrid method can be implemented thanks to information theory.

We have mentioned in chapter 2 that concepts directly linked to information theory — such as text compression algorithms — have already been used to tackle similar problems (*e.g.* [BD99, MRD04]). This is a sensible approach since good text compressors usually are good predictors. The method proposed in this chapter might at first seem very different from those works; the underlying concepts are however directly linked: the pieces of information that most reduce the entropy of the mobiles motion should be transmitted first.

5.2 Principle

First, let's give an overview of the framework studied here. We suppose that mobile nodes cross a cellular network and stay connected through *access routers*.

Some clues gathered during the motion of a MN can help guess its future movements. They can be collected by the MN itself or by ARs. Those routers can use them to predict MNs' next cell. The precise way those clues are generated and exchanged between MNs and ARs will be described in section 5.4.1.

Figure 5.1 shows a situation where each cell *(a)* monitors each MN's mean signal strength (SS) experienced during its journey and *(b)* compute each MN's cell travelling time (CTT). Those clues have a subscript describing to which cell they are related (*e.g.* CTT_1 denotes the time needed to cross cell 1). Mobiles moving along the railway have received a special '*train*' clue (when they enter a station, for example). The clues can be considered as random variables.

Cell 2 is crossed by 3 roads and a railway; it should figure out if a MN's next cell will be cell 3 or 4. This prediction can be done using CTT_1 alone: mobiles located on road 1 are characterised by a short travelling time. On the other hand,

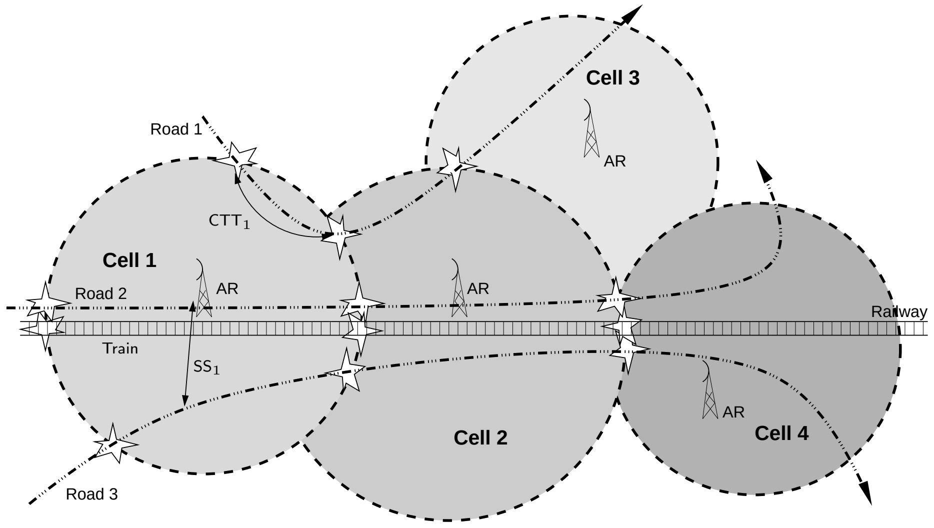


Figure 5.1: Overview.

the signal strengths SS_1 associated with roads 1 and 3 are about the same: this variable does not help much cell 2's prediction process.

Cell 4 must also guess if MNs passing by are located on road 2, road 3, or on the railway. Fortunately, this can be inferred from SS_1 (mobiles on road 3 are characterized by a weaker signal since this road is on the cell periphery) and the *train* clue (which discriminates between road 2 and the railway). Consequently, some variables should be propagated from one cell to another. For example, the *train* variable is certainly pertinent for all the cells along the railway; the same could be said about any variable that pinpoints the road a mobile is driving on.

We assume that MNs are responsible for sending variables from cell to cell. Since, in general, those terminals have strong memory and bandwidth constraints, we consider that the maximum number of clues it can hold is low. Other strategies could be conceived, such as asking the cells to broadcast data to neighbouring cells, or to rely on a MN's home repository, but both methods are likely to be more complex, consume bandwidth, and add delays.

This short example emphasizes two kinds of problems.

Problem 1: information estimation. Each cell has to estimate the relevance of each variable regarding next cell prediction. It is proposed to use the mutual information between the next cells and each variable as a relevance estimation. Mutual information is a direct measure of how the next cell prediction uncertainty is reduced thanks to the considered variable (section 5.3 gives a short reminder of information theory).

Problem 2: data spreading. Once —a reasonable approximation of— the information held by each variable for each cell is known, a sensible way of selecting the variables memorized by the MNs should be found. This selection scheme should be chosen so as to maximize the information gained by the cells along the mobiles paths.

One could argue that wireless terminals now have a comfortable amount of memory and could simply record all the collected clued, relevant or not. This calls for four comments. First, we will see that clues have to be exchanged regularly between the MN and ARs; thus, keeping unrelevant variables means wasting bandwidth. Second, this method should be fitted to mobiles' least common denominator: down-market terminals should not be omitted. Third, the amount of variables to consider could quickly grow since they could be frequently generated (*e.g.* GPS updates) and stay relevant for a long period of time. Finally, the technique presented here is expected to be applicable to other contexts and could therefore be considered a general "data routing" strategy aimed at maximizing data relevance.

5.3 Information theory and decision trees

Information theory ([vdL97]) and decision trees ([Qui93]) are building blocks of the work presented in the following sections. Those topics have already been introduced in section 2.1.3; they are thus only briefly recalled here for completeness.

5.3.1 Information theory

Consider a discrete random variable $\mathcal{X}$, its *entropy* $H(\mathcal{X})$ and its *conditional entropy* knowing $\mathcal{Y}$ $H(\mathcal{X}|\mathcal{Y})$ defined as:

$$H(\mathcal{X}) = -\sum_{x\in X} P(x)\ \log P(x) \tag{5.1}$$

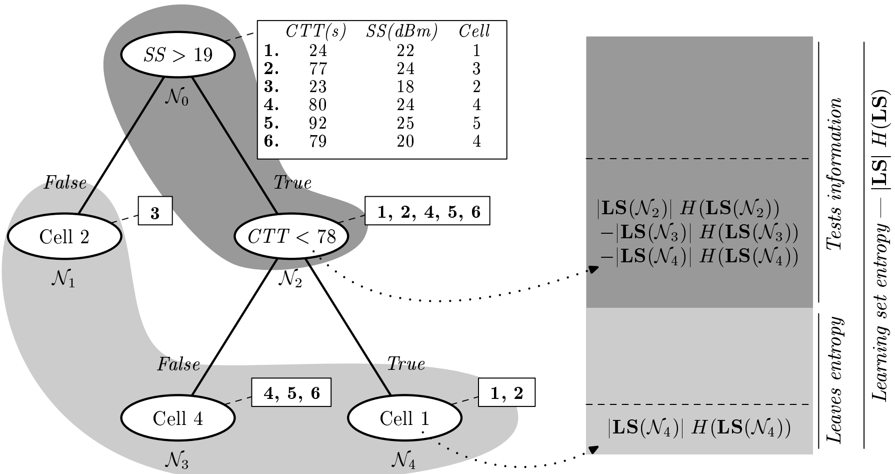
$$H(\mathcal{X}|\mathcal{Y}) = -\sum_{x\in X}\sum_{y\in Y} P(x,y)\ \log P(x|y) \tag{5.2}$$

where $P(x)$ is the probability of the event x, $P(x,y)$ the joint probability of x and y, $P(x|y)$ the probability of x given y, and X (resp. Y) the sample space of $\mathcal{X}$ (resp. $\mathcal{Y}$). In the following, the logarithm function is always supposed defined in base 2. The entropy quantifies the predictability of a random variable; an entropy equal to zero corresponds to a variable whose realizations values are known in advance.

The *mutual information* between variables $\mathcal{X}$ and $\mathcal{Y}$ is equal to:

$$I(\mathcal{X};\mathcal{Y}) = H(\mathcal{Y}) - H(\mathcal{Y}|\mathcal{X}) = I(\mathcal{Y};\mathcal{X}) \tag{5.3}$$

and measures the dependence between $\mathcal{X}$ and $\mathcal{Y}$ (*i.e.* the information gained on $\mathcal{Y}$ knowing $\mathcal{X}$; $I(\mathcal{X};\mathcal{Y}) = 0$ if $\mathcal{X}$ and $\mathcal{Y}$ are independent).



	CTT(s)	*SS(dBm)*	*Cell*
1.	24	22	1
2.	77	24	3
3.	23	18	2
4.	80	24	4
5.	92	25	5
6.	79	20	4

$SS > 19$
$\mathcal{N}_0$
False
True
Cell 2
$\mathcal{N}_1$
3
$CTT < 78$
$\mathcal{N}_2$
1, 2, 4, 5, 6
False
True
Cell 4
$\mathcal{N}_3$
4, 5, 6
Cell 1
$\mathcal{N}_4$
1, 2
$|\mathbf{LS}(\mathcal{N}_2)|\ H(\mathbf{LS}(\mathcal{N}_2))$
$-|\mathbf{LS}(\mathcal{N}_3)|\ H(\mathbf{LS}(\mathcal{N}_3))$
$-|\mathbf{LS}(\mathcal{N}_4)|\ H(\mathbf{LS}(\mathcal{N}_4))$
$|\mathbf{LS}(\mathcal{N}_4)|\ H(\mathbf{LS}(\mathcal{N}_4))$
Tests information
Leaves entropy
Learning set entropy — $|\mathbf{LS}|\ H(\mathbf{LS})$

Figure 5.2: A simple decision tree. This tree guesses a mobile's next cell using two measurements: a cell traveling time (CTT) and a signal strength (SS); those are the tree attributes $\mathcal{V}_1$ and $\mathcal{V}_2$. The $\mathcal{X}$ variable is the next cell and, consequently, the classes for this problem are cells 1 to 5.

5.3.2 Decision trees

A *decision tree* encodes a set of tests related to random variables (say $\mathcal{V}_1, \mathcal{V}_2, \dots$) as a tree. This tree aims at encoding a conditional probability law $P(\mathcal{X}|\mathcal{V}_1, \mathcal{V}_2, \dots)$ where $\mathcal{X}$ is a discrete random variable. A tree $\mathcal{T}$ defines a leaf distribution and each leaf T_i matches a probability distribution $P(\mathcal{X}|T_i)$.

The tree is built in such a way that the entropy $H(\mathcal{X}|\mathcal{T})$ is reduced as much as possible; this means that once the trees tests have been applied, the remaining uncertainty about $\mathcal{X}$ is small. The left part of figure 5.2 shows a simple decision tree.

In this context, variables' realizations are often denoted *samples*, $\mathcal{V}_i$ variables *attributes* and $\mathcal{X}$ the *goal attribute*.

We will see that the tree learning algorithm plays an important role in what follows. It is thus briefly described below[1].

The problem is to deduce the distribution $P(\mathcal{X}|\mathcal{V}_1, \mathcal{V}_2, \dots)$ given a set of realizations (or *learning set*, **LS**). A test is first chosen for the root node; this node has two sons, $\mathcal{N}_{\text{true}}$ and $\mathcal{N}_{\text{false}}$ corresponding the test's possible outcomes. The node $\mathcal{N}_{\text{true}}$ (resp. $\mathcal{N}_{\text{false}}$) matches a subset of the learning set —denoted $\mathbf{LS}(\mathcal{N}_{\text{true}})$ (resp. $\mathbf{LS}(\mathcal{N}_{\text{false}})$)— so that the test applied to the proper attribute of its elements is true (resp. false). The same procedure can be repeated for both root's son nodes.

Tree tests are chosen so as to maximize the information gained on the goal attribute; this quantity of information is estimated by subtracting the learning set

[1] This work could easily be adapted to other learning methods (*e.g.* [Fre95, Geu02]).

entropy once the test outcome is known to the entropy before the test is applied. Formally, the information brought by the test matching node $\mathcal{N}_\mathrm{f}$ is estimated by:

$$I(\mathcal{N}_\mathrm{f}) = H(\mathbf{LS}(\mathcal{N}_\mathrm{f})) - \frac{|\mathbf{LS}(\mathcal{N}_\mathrm{ls})|}{|\mathbf{LS}(\mathcal{N}_\mathrm{f})|} H(\mathbf{LS}(\mathcal{N}_\mathrm{ls})) - \frac{|\mathbf{LS}(\mathcal{N}_\mathrm{rs})|}{|\mathbf{LS}(\mathcal{N}_\mathrm{f})|} H(\mathbf{LS}(\mathcal{N}_\mathrm{rs})) \qquad (5.4)$$

where $\mathcal{N}_\mathrm{ls}$ and $\mathcal{N}_\mathrm{rs}$ are the left and right sons of $\mathcal{N}_\mathrm{f}$ and where $|\mathbf{S}|$ denotes the cardinality of set $\mathbf{S}$. A set of samples' entropy is estimated using (5.1) applied to the estimated distribution of $\mathcal{X}$ for this set.

The tree is recursively expanded until a pruning criterion is met (refer to [Qui93] for in-depth explanations).

An interesting property links the tests information, the learning set entropy and the leaves entropy: the sum of the weighted tests information (as estimated by equation 5.4) is equal to the learning set entropy ($|\mathbf{LS}|H(\mathbf{LS})$) minus the weighted entropy of the samples sets associated with leaves (as given by equation 5.1). The weights are given by the number of samples matching the corresponding node.

This section is summarized in figure 5.2. It shows a small tree that could be used in a cellular network to guess in which cell a mobile might be going given the time it took to cross the cell (its cell traveling time, CTT) and a measured signal strength (SS). This tree has been pruned ($\mathcal{N}_3$ and $\mathcal{N}_4$ might have been split).

The right-hand side of the figure depicts that the learning set entropy can be divided into two parts: the tests information (dark grey) and the leaves entropy (light grey). The formulas needed to compute the information gained in $\mathcal{N}_2$ and the entropy of $\mathcal{N}_4$ —which are particular cases of (5.4) and (5.1)— are explicitly given. This example tree is quite inefficient: the information collected by its tests is only two thirds of the learning set entropy.

Each leaf $\mathcal{N}_l$ can be labelled with a *class*, *i.e.* the most frequent value of the $\mathcal{X}$ attribute for the set $\mathbf{LS}(\mathcal{N}_l)$.

Sample classification (*i.e.* finding a sample's most likely $\mathcal{X}$ attribute value) can be done efficiently. It only requires to cross the tree from its root down to a leaf according to the outcomes of the nodes' tests applied to the sample. The complexity of tree building is not considered a critical issue since this operation is not supposed to occur frequently[2].

5.4 Mobility prediction

5.4.1 Principle

We suppose that MNs travel a cellular network and collect clues related to their motion. Those clues can be generated by the network (*e.g.* a cell identifier) or by

[2] The complexity of the tree building algorithm is $KN \log N$ where K is the number of attributes and N the cardinality of the training set.

the mobile terminal itself (*e.g.* a received signal strength measurement). Since, in general, MNs are small terminals, we consider that the maximum number of clues it can hold is low. The problem is thus to find which clues are worth keeping.

Figure 5.3 shows how the clues can be handled. We consider that MNs send these pieces of information to an AR which is responsible for analyzing them so as to guess the terminal's next cell (*i.e.* collecting them in a *samples database* as depicted figure 5.2 and building a decision tree). Clues can be considered by the ARs as realizations of random variables likely to give information on the random variable matching the *a priori* distribution of possible next cells.

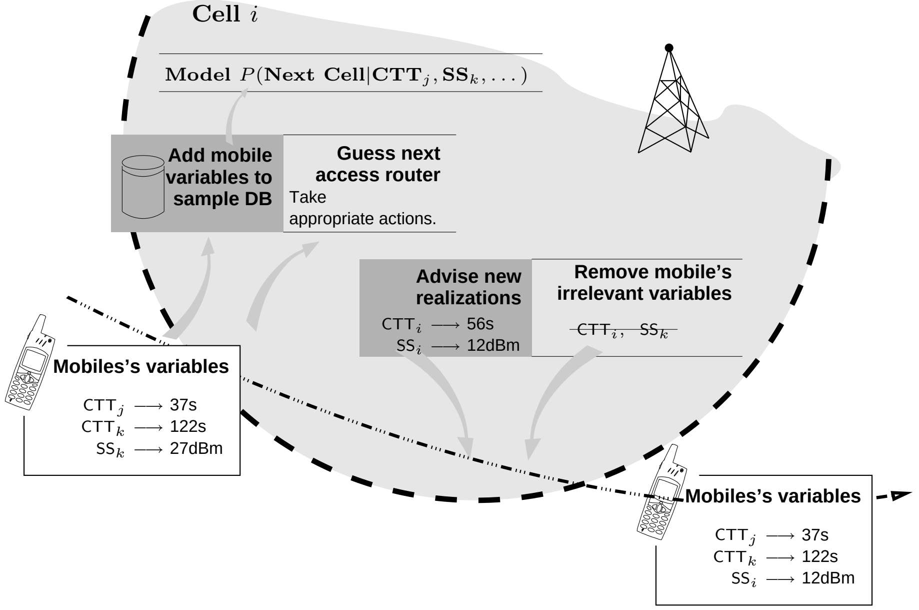


Figure 5.3: Mobility prediction overview. Each cell performs a variable selection. Variables names have a subscript describing where they have been generated; thus, CTT_j and SS_i represent the time taken to cross cell j and a signal strength measurement done in cell i.

Notice that the memory and processing consuming tasks (*i.e.* collecting a samples database and using it to predict a mobile's destination) are left to the ARs: the mobile acts as a dumb terminal.

The ARs use the mobile's variables to guess its probable next cell, and then take appropriate actions (such as resources reservation). This requires the samples database to be populated with the actual next cells reached by the mobiles; this can be solved by asking all the cells to warn the previous cell each time a handoff occurs.

During the mobile's cell crossing, the AR generates new variables that can be added to those the mobile already holds. Since the MN's amount of memory is limited, the AR asks to remove variables considered not relevant enough.

The next section is dedicated to next cell prediction. The following two deal with problems we have already mentioned: determining the information received from a variable realization and choosing which variable should a MN keep or remove.

5.4.2 Next cell prediction

The samples database collected by an AR can be represented as a table where each row is a sample and each column matches a variable; we denote $\mathcal{A}_k$ the set of those variables for cell k. This set holds all the variables that have already been observed.

All the mobiles entering a given cell will not hold the same variables (since it is likely that they did not cross the same cells). The variables that belong to $\mathcal{A}_k$ but not to the set of variables held by an incoming mobile are assigned the special *notApplicable* value. This value should be easy to discriminated from the other realizations by the tree (*e.g.* a negative number if the variable domain is positive numbers).

An AR can build a decision tree as soon as the database is considered populated enough. Cell k's tree is denoted $\mathcal{T}_k$; notice that it models the probability distribution of next cells given the mobile clues. Next cell prediction just amount to classify the sample made of the MN's variables using the tree.

5.4.3 Information estimation

5.4.3.1 Principle

Once a mobile leaves a cell, the AR should filter its variables and remove those expected to be less useful to the MN's next cells. Section 5.4.4 will show that this requires that each cell estimates how useful a variable is to its own prediction (*i.e.* the information it gives regarding next cell guessing).

This can be estimated thanks to the decision tree introduced in the previous section. We have seen that the information $I(\mathcal{N})$ of each test node $\mathcal{N}$ composing the tree can be estimated by (5.4); a way to evaluate the information related to an attribute A is thus to sum the information of all the tree's tests involving A. This leads directly to the following formula:

$$I(A) = \sum_{\text{Test node } \mathcal{N}} \frac{|\mathbf{LS}(\mathcal{N})| \cdot I(\mathcal{N})}{|\mathbf{LS}|} \tag{5.5}$$

where the sum operates over the nodes testing A.

5.4.3.2 Example

Figure 5.4 gives a simple application of (5.5). A mobile speed and angle of arrival are used to figure out on which road it is located. There are three roads, each with a specific mean speed (80, 90 or 100 km/h) and angle of arrival (160, 170 or 180°). The speed and angle values are distributed according to:

$$\begin{aligned} V_{\text{speed}} &= \overline{V_{\text{speed}}} + \alpha \cdot N(0; 25) \\ V_{\text{AoA}} &= \overline{V_{\text{AoA}}} + (1 - \alpha) \cdot N(0; 15) \end{aligned}$$

where $N(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ. The α factor is a constant between 0 and 1 describing if the Gaussian noise is added to one variable or the other. The first two plots of figure 5.4 depicts those distributions. The roads are equally populated, yielding an *a priori* system entropy of ~ 1.6 bits ($\log\ 3$).

The information theory related values we have discussed so far have been estimated using decision trees built for various values of α (bottom of figure 5.4). When α is nearly equal to zero (resp. one), the system entropy is entirely compensated by the information given by the speed (resp. angle) variable. Notice that the attribute's information curves cross each other when $\alpha \approx .37$ (not .5) since the variables' standard deviations are different. At this moment, the curves have a small gap because the tree's root node's attribute changes; the algorithm presented here over-estimates the information brought by attributes near the tree's root. Thus, in practice, it cannot be assumed that equally meaningful attributes will receive the same information measurement.

Prediction success ratios have been estimated with the learning set (*i.e.* classifying the samples used to build the tree) and with an independent test set. The resulting curves are nearly merged, showing that no over-learning (leading to over-estimated variable information) has occurred. The sum of the entropy of each trees' leaves has been plotted. This sum shows the remaining uncertainty about the samples' class once all the tree's tests have been applied. As expected, the success ratio is low when the leaves entropy is high.

5.4.3.3 Biasing the tree building

When two variables $\mathcal{V}_1, \mathcal{V}_2$ hold the same information regarding $\mathcal{X}$ (*i.e.* $I(\mathcal{X}; \mathcal{V}_1) \approx I(\mathcal{X}; \mathcal{V}_2) \approx I(\mathcal{X}; \mathcal{V}_1, \mathcal{V}_2)$), (5.5) tends to divide the information $I(\mathcal{X}; \mathcal{V}_1, \mathcal{V}_2)$ evenly among $\mathcal{V}_1$ and $\mathcal{V}_2$. Unfortunately, as we will see later, it would be preferable to clearly differentiate those variables and to associate a high information with one of them —and a low information with the other.

To get this behaviour, we slightly change the way node tests are chosen while building a tree. We have said that tests are selected so as to maximize the test information given by (5.4); we propose to slightly change this behaviour. A test is chosen in the set of tests T such that:

- it contains the test t that maximize (5.4), let $I_{\max}$ be t's information;

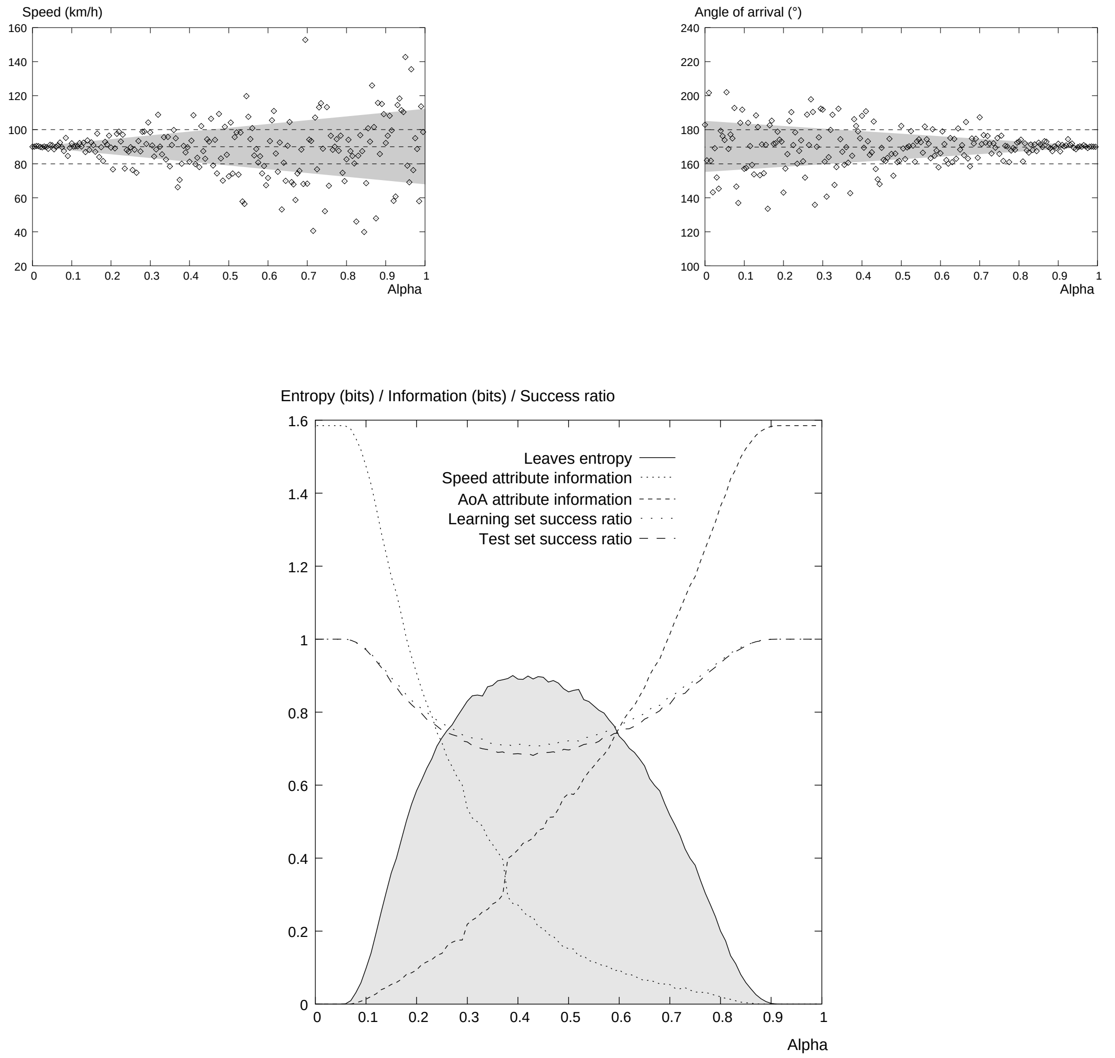


Figure 5.4: Attributes information. *Top:* distributions of the speed (left) and angle of arrival (right) variables. The dashed lines depict the mean values for each road and the grey zone corresponds to the distributions' standard deviations. *Bottom:* various values extracted from the decision trees for different values of the $\boldsymbol{\alpha}$ parameter.

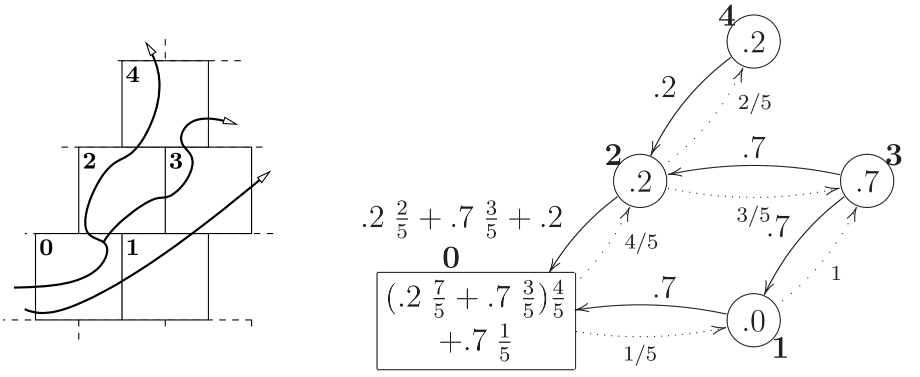


Figure 5.5: Principle of the information spreading mechanism. Dotted arrows depict the mobiles movement; their labels give the value of the $f_{\mathcal{V}}$ functions. Plain arrows show the expected information propagation back to cell 0.

- the difference between $I_{\max}$ and the information of the other tests in T is smaller than a constant chosen equal to 0.01 bit[3].

The "first" test among the elements of T is finally chosen using an arbitrary order relation defined on the tests' attributes *a priori*. This way, the problem mentioned above is mitigated since, when it is reasonable, the same variable is chosen for most tree nodes.

5.4.4 Data spreading

The previous section shows how to estimate the information carried by a variable *for a specific cell*. Our aim is to maximize the information collected by the cells during the mobile journey, so we need *(a)* to let a cell figure out the expected information brought by an attribute along a mobile's future path and *(b)* to deduce from that information which attributes should be carried by a mobile.

In the following, $\mathcal{V}_i^j$ will denote cell j's i-th variable and V_i^j its realization. For example, the variables generated by cell i in figure 5.3, CTT_i and SS_i, might be denoted $\mathcal{V}_0^i$ and $\mathcal{V}_1^i$.

5.4.4.1 Attribute's expected information

We try to guess how much information can the $\mathcal{V}$ variable give to the cells on a MN's path.

We first consider an AR that has never seen any MN leaving its cell with $\mathcal{V}$ in its attributes set. Obviously, $\mathcal{V}$ will not give any information to the MN's next

[3] Simulations results have shown to be insensible to the actual value of this parameter as long as it is chosen small enough.

cells and the expected information for $\mathcal{V}$ amounts to the information given to the current cell (see procedure `spread`, lines 2–5 —the `treeInfo` procedure is given by (5.5)).

Algorithm 5.1: spread($\mathcal{V}_i^j$) Information spreading of $\mathcal{V}_i^j$ for cell k

```
1  var $\overline{I}_{\mathcal{V}_i^j} : \mathbf{N}_{\mathcal{V}_i^j} \rightarrow ]0; 1]$;
2  if $\mathbf{N}_{\mathcal{V}_i^j} = \emptyset$ and $j \neq k$ then
3      $\forall p \in \mathbf{P}_{\mathcal{V}_i^j}$ : send treeInfo($\mathcal{T}_k, \mathcal{V}_i^j$) to $p$;
4      return;
5  end
6  $\forall n \in \mathbf{N}_{\mathcal{V}_i^j}$: receive $\overline{I}_{\mathcal{V}_i^j}(n)$;
7  if $\mathbf{N}_{\mathcal{V}_i^j} \neq \emptyset$ and $j \neq k$ then
8      $\forall p \in \mathbf{P}_{\mathcal{V}_i^j}$ : send treeInfo($\mathcal{T}_k, \mathcal{V}_i^j$) +
9          $\sum_{n \in \mathbf{N}_{\mathcal{V}_i^j}} \overline{I}_{\mathcal{V}_i^j}(n) \cdot f_{\mathcal{V}_i^j}(n)$ to $p$;
10 end
11 return $\overline{I}_{\mathcal{V}_i^j}(\cdot)$
```

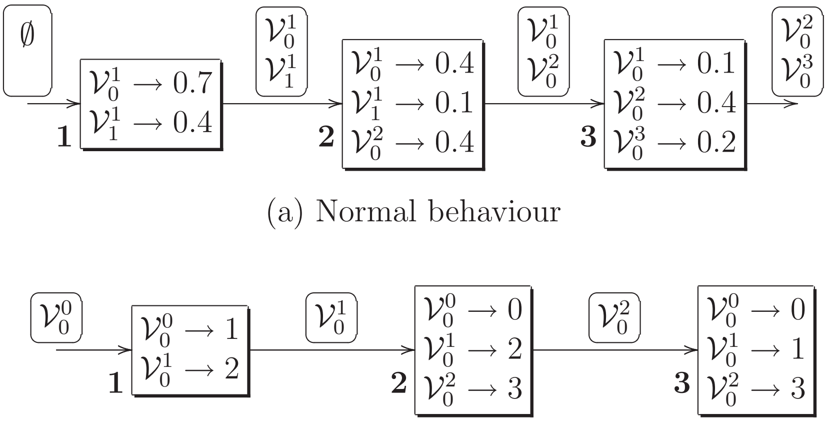
This expected information can be sent to the *previous cells*, *i.e.* the cells that have sent at least one mobile holding $\mathcal{V}$; the set of previous cells is denoted $\mathbf{P}_{\mathcal{V}}$. Similarly, the *next cells* set, $\mathbf{N}_{\mathcal{V}}$, contains the cells where at least one mobile holding $\mathcal{V}$ has gone.

It is likely that MNs will not visit cells in $\mathbf{N}_{\mathcal{V}}$ homogeneously. We denote $f_{\mathcal{V}}$ the function $\mathbf{N}_{\mathcal{V}} \rightarrow]0; 1]$ such that $f_{\mathcal{V}}(n)$ is the ratio of mobiles holding $\mathcal{V}$ leaving the current cell for cell n ($\sum_{n \in \mathbf{N}_{\mathcal{V}}} f_{\mathcal{V}}(n) = 1$).

Once an AR knows, for all its next cells, the expected information received along a mobile's path starting with cell n —denoted $\overline{I}_{\mathcal{V}}(n)$—, it can compute its own information expectation using *(a)* the information given by $\mathcal{V}$ *(b)* the values $\overline{I}_{\mathcal{V}}(n)$ weighted by $f_{\mathcal{V}}(n)$; this is summarized in the `spread` procedure, line 8.

Figure 5.5 gives an overview of the information spreading process for a variable emitted by cell 0. The top part shows 5 cells crossed by a few roads. The bottom part depicts the spreading algorithm applied to this simple topology. It shows the local information as computed by equation 5.5 (circled numbers), $\mathbf{N}_{\mathcal{V}}$ sets (dotted arrows), the corresponding $f_{\mathcal{V}}$ functions (dotted arrows' labels) and the information estimation propagation from cells 3 and 4 back to cell 0 (plain arrows).

Notice that the algorithm presented here does not deal with cycles: the motion graph is supposed acyclic. Even if this should not cause any real impact in practice, it would be desirable to modify it to get rid of that limitation.



(a) Normal behaviour

(b) A catch

Figure 5.6: Information spreading. The bold rectangles give the variables expected information for the next cells.

5.4.4.2 Attributes selection

In the following, we assume that every MN can stock a fixed, finite number of attributes. Once all the cells have learned their $\overline{I}_{\mathcal{V}_i^j}$ functions (one function for each variable that might leave the cell), the network's ARs can efficiently select a MN's most pertinent variables.

Ideally, this selection should be such that the expected sum of the MNs' variables information for all the visited cells is maximized; we call it the *optimum selection scheme.*

Since the $\overline{I}_{\mathcal{V}_i^j}$ functions are known, the straightforward wait to select m variables out of a variables set V is to sort them by decreasing information expectation and pick up the first m. A small improvement of this algorithm is to first guess the MN's next cell n so as to use $\overline{I}_{\mathcal{V}}(n)$ to measure the usefulness of $\mathcal{V}$.

Figure 5.6 *(a)* shows the normal behaviour of this algorithm. In this example, a MN handles 2 attributes (rounded rectangles) and crosses 3 cells laid out linearly. The bold rectangles show the variables' expected information (*e.g.* in cell 1, $\overline{I}_{\mathcal{V}_0^1}(2) = 0.7$); one can verify that variables are selected as explained above. Notice that on that simple topology, the variables information have to decrease from a cell to the next, the difference being the information gained locally (*e.g.* since $\mathcal{V}_0^1$ drops from 0.7 in cell 1 to 0.4 in cell 2, 0.3 bits are local to cell 2).

This is where the observation formulated section 5.4.3.3 is relevant: it helps get very contrasted variables lists with very relevant and very irrelevant variables and few variables in between.

Figure 5.6 *(b)* demonstrates that this simple solution is not optimal. Cell 1 chooses to send $\mathcal{V}_0^1$, but this variable is immediately replaced by $\mathcal{V}_0^2$ in cell 2. The

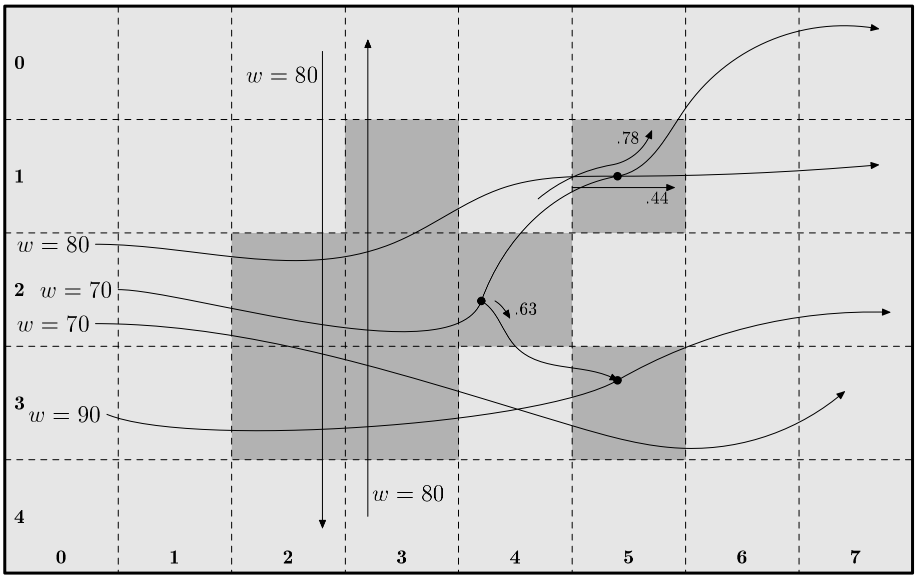


Figure 5.7: The simulation setup.

attribute $\mathcal{V}_0^1$ does not give any information to cell 2, but $\mathcal{V}_0^0$ does: it was the best variable to send between cell 1 and 2. The algorithm is thus a victim of its greediness.

5.5 Simulation

5.5.1 Simulation setup

The work presented above has been tested on a simulation. A grid made of 8x5 cells composes the simulation setup (figure 5.7).

Two roads let the MNs cross the map vertically. Five roads allow the mobiles to cross the terrain from left to right. If the roads had been bidirectional, inferring a MN's direction would have required to know one of its previous cells. Since every variable is implicitly related to the cell that generated it, knowing the mobile's direction would have been an easy task. It has thus been decided to use a simpler configuration with unidirectional roads since bidirectional ones would not have changed the simulations results significantly.

Each cell generates two kinds of variables: the MN's *cell travelling time* (CTT) and a *noise* variable with randomly generated realizations. The latter demonstrates the ability of the system to differentiate between relevant and less relevant variables. One should note that this *noise* variable can nevertheless bring some information since a noise variable indicates at least that the mobile has crossed the cell that generated it; this might give a piece of information about its future path.

One third of the MNs are created with a special *emergency* variable. This variable flags mobiles that are certainly going to cell (7,1). This shows that variables could be associated with destinations when the mobile path is known in advance; this emergency variable could be associated with emergency vehicles that are very likely to go to the nearest hospital or police station. Another possibility would be to flag mobiles entering a train.

Each road is labelled with a weight w that indicates its significance; the higher this number, the faster the cell is crossed and the higher the number of mobiles using it. The cell travelling time is related to the road weight w by the relation $\text{CTT} = 120 - 30\ w/100 + N(0;5)$.

Crossroads are depicted with a black dot. When applicable, an arrow gives the ratio of mobiles going in each direction.

Dark grey cells are those where mobility prediction is needed: the next cell of MNs leaving those cells is not known in advance.

Three variable selection methods have been compared. The first is the most obvious scheme where the oldest variable is removed first; this is equivalent to putting variables in a bounded FIFO queue. The second method is based on decision trees; each cell learns the parameters of a tree and use it to measure variables relevance. The less relevant variables are removed first. The cells do not exchange variable relevance estimations with each other. The rational behind this scheme is that if a cell finds a variable interesting, it is likely that it will also be relevant for its neighbours. Those schemes have been compared to the entropy-based scheme explained above.

Notice that those three methods differ only in the way the relevant variables are chosen. A direct consequence is that the amount of information exchanged between the mobiles and the ARs is the same; the only difference resides in the processing required by the ARs.

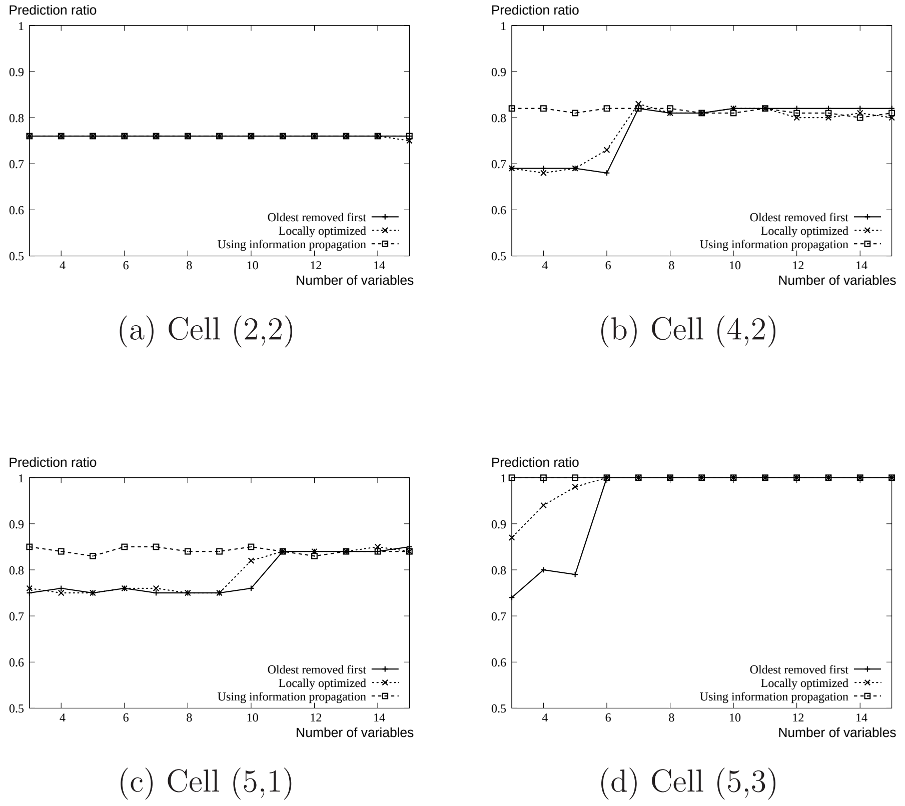
All the mobiles can carry the same fixed number of variables; this number is a simulation parameter. Its influence on the next cell prediction ratio has been quantified.

5.5.2 Results

Figure 5.8 shows, for 4 different cells, the ratio between the number of correct next cell predictions and the total number of predictions

The first plot is related to cell $(2, 2)$; it shows that all the methods perform equally. This is not surprising since mobiles are crossing the map from left to right: variable selection can only happen after a few cells have been crossed, so the interesting cells are on the right-hand side of the map.

Thus, the other plots are related to cells (4,2), (5,1) and (5,3); they allow to draw several conclusions. On this simple cell topology, the information-based method



(a) Cell (2,2)

(b) Cell (4,2)

(c) Cell (5,1)

(d) Cell (5,3)

Figure 5.8: Simulation results for the 4 cells depicted in figure 5.7.

gives an optimum result with as few as 3 variables[4]. When the MNs' memory is low, the other methods always gives lower prediction ratios. The number of variables required to catch up varies from cell to cell but it can be as high as 11, which is more than one would expect for such a small map.

The improvement of the *locally optimized* method over the simple, *older-first* scheme does not seem to be worth the increased implementation complexity. This shows that the upstream propagation of variable relevance measurements is needed: variable significance is not a local property.

Even if they should be confirmed on a large scale experiment, those results are encouraging since the method's usefulness is expected to increase with the number of cells — and, thus, the number of potential variables — encountered. For instance, an attribute created when a mobile user enters a train will be useful for tens of cells along the railroad; the data spreading paradigm should allow to bring this observation to the fore.

5.6 Long term predictions

In some circumstances, it is possible to guess not only one, but several cells the mobile will eventually cross. Such long-term predictions can be achieved accurately when, for example, mobiles are moving in a train or on a highway and can be used, for example, to lower the number of location updates needed to perform paging.

A terminal destination can also be determined thanks to the value of one (or several) random variables realization(s) — such as the *emergency* variable introduced in section 5.5. The nature of those variables lets them give information to a large number of cells, increasing the probability of being picked up by the cells on the mobile path thanks to the information spreading method. Such variables should allow to determine which cells the mobile will go through to reach its destination.

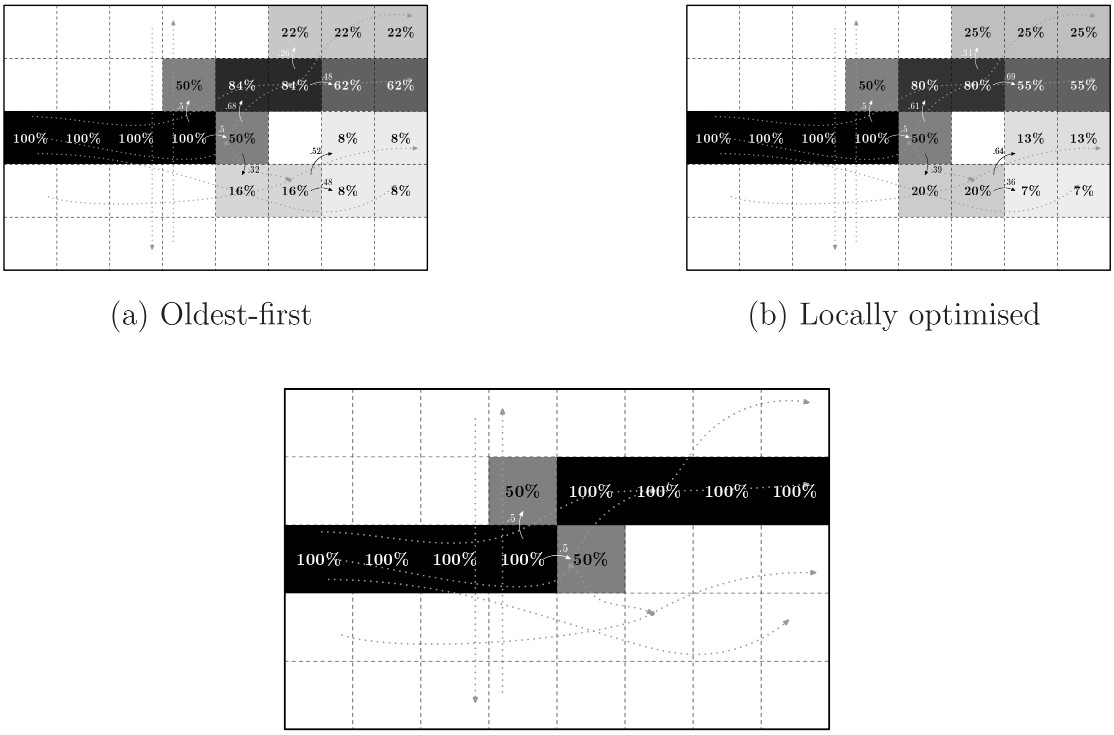
The following explains how long-term predictions can be obtained and, using simulations, shows how it performs.

The learning and information spreading techniques presented in section 5.3 and 5.4 do not have to be modified, thus we assume that the expected information brought by the variables has been computed and is known to every cell.

When a mobile enters a cell, the set of variable realizations V it brings can be used to predict its set of potential next cells $C = \{c_i\}$ and the associated probabilities $p_C : C \rightarrow [0, 1]$. Those probabilities are directly derived from $\mathbf{LS}(\mathcal{N}_l)$ where $\mathcal{N}_l$ is the decision tree's leaf corresponding to V.

The current cell can apply the variables selection algorithm as before (section 5.4.4.2), but does not know what the realizations of the variables added by the cell will be when the mobile performs its handoff. Those realizations are thus

[4] Experiments using an arbitrary large number of variables have shown that the prediction ratio cannot be increased further.



(a) Oldest-first

(b) Locally optimised

(c) Information-based

Figure 5.9: Simulation results of long-term predictions using different information dissemination methods.

replaced by *unknown values* to produce a new V_C vector. To classify a sample with unknown values, a tree is allowed to explore both possible outcomes of a test and to average the results; the interested reader can refer to [Qui93] for a detailed explanation.

This vector and the probability $p_C(c_i)$ can be sent to the potential next cell c_i. c_i repeats the same procedure to compute a vector $V_{C'}$ and to get the mobile's potential next cells $C' = \{c'_i\}$ and the probabilities function $p_{C'} : C' \rightarrow [0, 1]$. The product $p_C(c_i) \cdot p_{C'}(c'_j)$ gives the probability that the mobile crosses cells c_i and c'_j. This process can be repeated until the product $P_{c_i,c'_j,c''_k\dots} = p_C(c_i) \cdot p_{C'}(c'_j) \cdot p_{C''}(c''_k) \dots$ is small enough.

The probability to cross a given cell c is given by the sum of the products relative to a list of cells terminated with c (such as $P_{c_i,c'_j,\dots c}$). Figure 5.9 gives this probability computed for the cells of the topology presented in the previous section. The results shown have been obtained for a mobile using the *emergency* variable, which, as explained before, is a good candidate for long-term prediction.

As expected, both the *oldest-first* and *local* methods only predict the first four cells the mobile will cross. Using the *information-based* method, the *emergency* variable is kept throughout the MN's journey since it gives information to several cells, increasing its probability to be picked up by the `select` algorithm.

Notice that this approach applied to the problem of paging would yield a method where the most probable cells could be paged first; this result is similar to what could be achieved using the work presented in [BD99].

5.7 Future work

This work can be extended along several lines:

1. The network has been considered static; this work could be adapted to deal with changing probabilities, where new variables are added and the information associated with a variable can vary with time.

2. The spreading algorithm could be improved; in particular, it is desirable to compute the *optimum selection scheme* (or a reasonable approximation if the problem is proven NP-complete, see section 5.4.4.2).

3. The simulations shown here involve a small map and a small number of variables. It would be desirable to show how the method performs with real-scale scenarios and real-world data.

4. The method presented in this paper is also expected to be applicable to other contexts than mobility prediction. Since it is well suited to situations where sending information is costly, it could thus be applied to sensor networks where both low available energy and low bandwidth are reasons to transmit as few data as possible. The proposed approach could select the most relevant sensors among all those that have been scattered at a given place. This work could also help route relevant pieces of information in the so-called *Knowledge Plane* proposed by Clark et al., see [CPRW03].

5.8 Conclusions

We have proposed a method for sending pieces of mobility prediction information where it is most needed. Knowing which clues are useful is done by estimating their expected information on the upcoming path of each mobile.

Simulations have shown that the method allows one to discriminate relevant clues, to send them where they are needed in the network and, consequently, to improve mobility prediction.

It has been shown that when a clue is pertinent for a large number of cells, it increases its probability of staying in the network and allows one to predict several next cells in advance.

This work is expected to be applicable to other contexts involving constraints such as scarce bandwidth and pieces of information of varied relevance. Sensor networks for example seem good candidates.

Part III

Using Mobility Prediction

6 Improving Call Admission using Mobility Prediction

There is no absolute knowledge. And those who claim it, whether they are scientists or dogmatists, open the door to tragedy.
All information is imperfect. We have to treat it with humility.

— JACOB BRONOWSKI, *The Ascent of Man* (1973)

We have seen in chapter 2 that mobility prediction can help reserve resources proactively. As in section 2.2.4, we study a cellular network where some bandwidth can be reserved in advance in order to decrease the probability of handover blocking.

Instead of studying a particular prediction scheme, we here replace the prediction process with an *abstract mobility prediction model* that allows us to observe the impact of various parameters such as the correct prediction ratio.

Using simulations, we show that each mobile has an incentive to act selfishly, and we propose a mechanism to promote fairness.

6.1 Introduction

We have already explored in chapter 2 the variety of mobility prediction scheme that have been proposed in the scientific literature. Since they are based on different assumptions, they give very varied results in terms of precision and quality.

Admission control and prediction are directly related since a frequent admission control requirement is to reserve resources so as to improve the network's future QoS; the amount of resources to reserve at a given time thus depends on the users upcoming behaviour.

This diversity leads to a situation where various (sometimes similar) admission control algorithms are simulated with very different prediction schemes.

Estimating the added value brought by a more precise prediction method is not an easy task; to which extent is it interesting to implement a more reliable (and often more complex) method? This chapter aims at answering this question in the particular context of call admission control.

We will see that this analysis leads to an improved admission method aimed at improving both network performance and fairness; this admission method can be implemented in a simple or in a more complicated way, yielding results in accordance with the added complexity.

6.2 Call Admission Control and mobility prediction

6.2.1 Overview

Admission control is tightly linked to mobility prediction, because reservations must be done in advance and at the right moment ([PGM04]). This prediction is usually done statistically: mobiles' movement behaviours (*e.g.* mean cell travelling time, or most likely next ARs) and call habits are analysed and averaged (*e.g.* [LAN95, CS98b]), taking care of their possible variations during the day and throughout the week.

Others use simple ([LSG01, YHP02]) or more advanced ([CB00, SK01, SK03, SK04]) techniques to track the individual motion of each mobile, hence getting an accurate idea of when and where each of them will handoff. A natural way to tackle this issue is to estimate each mobile's position (*e.g.* [LBC98]), but other methods can be considered (*e.g.* [FLM04]).

A mobile's next cell can also be inferred from its typical travelled cells; the prediction then amounts to guess the next cell given the previous ones. This can be achieved by various methods such as information theory ([BD99, RDM04]) or Markov chains ([LM95]).

6.2.2 An abstract model

The following aims at showing how prediction's performance influences CAC's effectiveness without restricting ourselves to a specific prediction mechanism. It has been chosen to model any mechanism with an abstract model defined by a few parameters. It is depicted in figure 6.1.

With a few values, this simple model summarizes the main information brought by a prediction algorithm. Those parameters should have an intuitive meaning and act upon the behaviour of the method using predicted events. At each trigger, the prediction mechanism reports an identifier of the next AR and the estimated delay before the next handover (δ seconds). This last measurement is optional as some schemes have not been designed to give this information. The trigger-to-handover delay is only estimated by the prediction algorithm, so the difference between this estimation and the real delay can be seen as realizations of a random distribution; let σ be the standard deviation of this distribution.

As one can see, this abstraction is only a rough sketch and several real systems' characteristics have been ignored, mainly:

- At most one trigger can be emitted before a given handover;
- The prediction success ratio is fixed once for all; one cannot model behaviours such as an increasing prediction success ratio when the trigger is delayed.
- Each trigger only gives the most likely next AR (it could guess the n next ARs, or several ARs with their probability of being next);
- The only information given about the trigger-to-handover duration are its mean value and standard deviation. Nothing more is given about the duration distribution which hence will most probably be chosen gaussian.

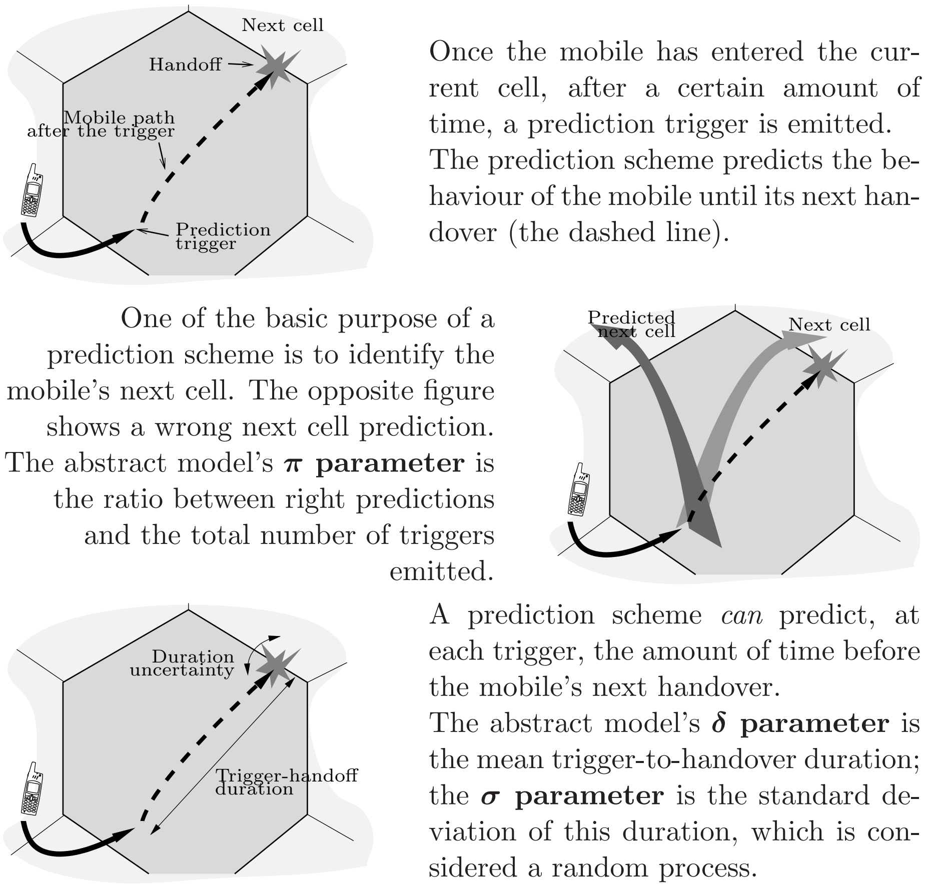
Let's now study the effect of those parameters on a concrete CAC algorithm.

6.3 The PCR algorithm

This CAC algorithm has been introduced in [CB00] by M.H. Chiu and M. Bassiouni; it is a typical CAC using mobiles' movements guesses extensively. Other methods are however possible, such as the one presented by Zhang and Liu (see [ZL01]) which keep the handover blocking probability under a given threshold using guard channels; the number of guard channels is modified adaptively. Its aim is to reduce the handover blocking probability in a cellular network by means of reserved channels. It is briefly described below; the interested reader can refer to [CB00] for more information.

Several mobiles are moving in a wireless network made of cells having a fixed number of channels. All the cells are assumed to have the same number of channels, and a given mobile can only use one channel at a time. The effect of soft handovers, frequency borrowing or space diversity techniques is not studied here. This implies that the proposed CAC scheme quite well matches 2G GSM but should be reevaluated in order to be applied to 3G networks ([SOH02, Che03, BH04, KMR04]).

Each mobile position is monitored so as to extrapolate its next cell, C. If a mobile has an ongoing call, it reserves a channel of C; if there is no free channel, the reservation is pushed on a unbounded FIFO reservation queue (a reservation is popped out every time a channel is freed until the queue is empty). When a mobile



Once the mobile has entered the current cell, after a certain amount of time, a prediction trigger is emitted. The prediction scheme predicts the behaviour of the mobile until its next handover (the dashed line).

One of the basic purpose of a prediction scheme is to identify the mobile's next cell. The opposite figure shows a wrong next cell prediction. The abstract model's $\boldsymbol{\pi}$ **parameter** is the ratio between right predictions and the total number of triggers emitted.

A prediction scheme *can* predict, at each trigger, the amount of time before the mobile's next handover.
The abstract model's $\boldsymbol{\delta}$ **parameter** is the mean trigger-to-handover duration; the $\boldsymbol{\sigma}$ **parameter** is the standard deviation of this duration, which is considered a random process.

Figure 6.1: This figure depicts a simple abstract model that modelizes the main characteristics of a mobility prediction scheme.

wants to start a new call, it has to find a free (unreserved) channel in its current cell; if there is none, the call is rejected. A mobile can of course cancel a previous reservation if it finds out that it will be useless (because of a sudden motion change or because its current call has ended).

[CB00] proposes three different ways to deal with handovers, called PCR1, PRC2 and PCR3. We won't study the intermediate algorithm PCR2 here, but focus on extreme cases, explained below. When a handover occurs:

PCR1	If the call has a prior reservation, it is allocated. Else, if there is a free channel, it is allocated. Else, the call is dropped.
PCR3	If there is a reserved channel, it is allocated. Else, if there is a free channel, it is allocated. Else, the call is dropped.

As one can see, the difference is that channels can be reserved either for a particular mobile (in that case, every reservation holds a mobile identifier) or "anony-

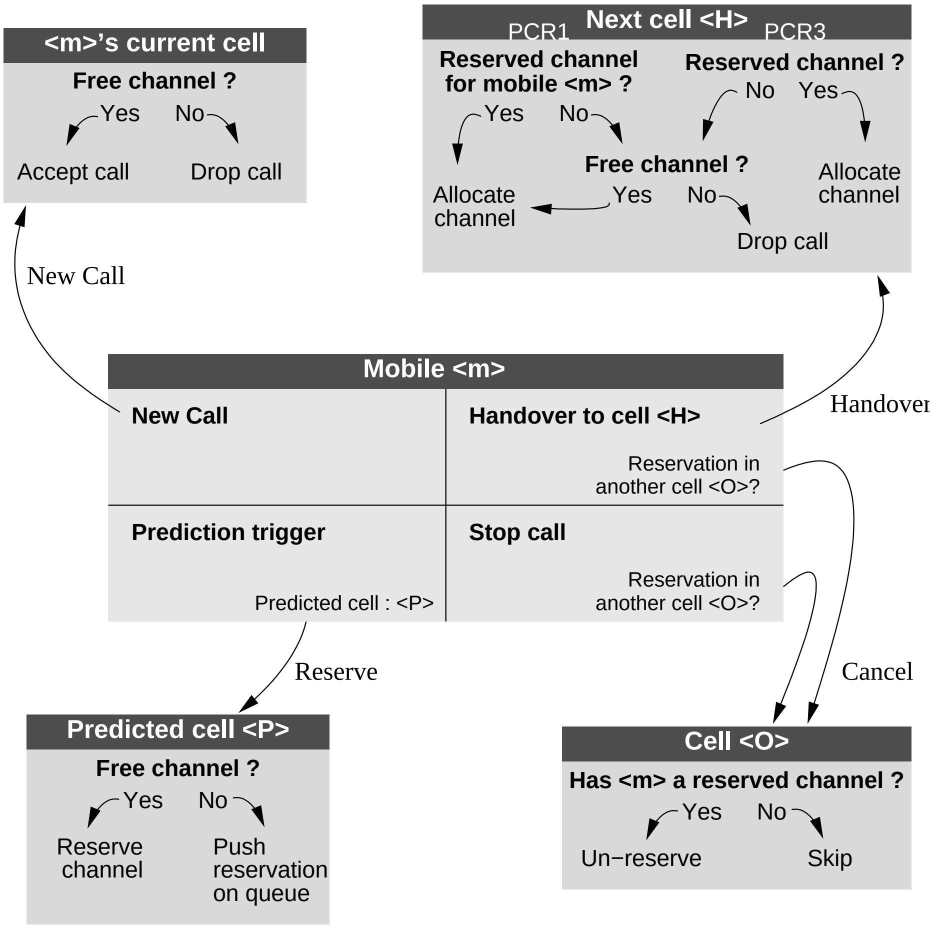


Figure 6.2: The PCR algorithm. This figure depicts the way PCR1 and PCR3 algorithms operate and how they can interoperate with an abstract mobility prediction scheme.

mously". This algorithm can be studied using the abstract prediction scheme introducted before. Note that as this scheme only allows one prediction per handover, the only way to notice a wrong prediction is to wait for the actual handover (and to see if it occurs with the predicted cell). The PCR algorithm and the way it has been mixed with the abstract prediction scheme have been summarized in figure 6.2.

6.4 Simulations

We can now study how PCR behaves according to the parameters of the prediction scheme. The aim is to find what characterizes a *good* predictive algorithm (in the CAC context) and to what extent its performance can modify those of PCR.

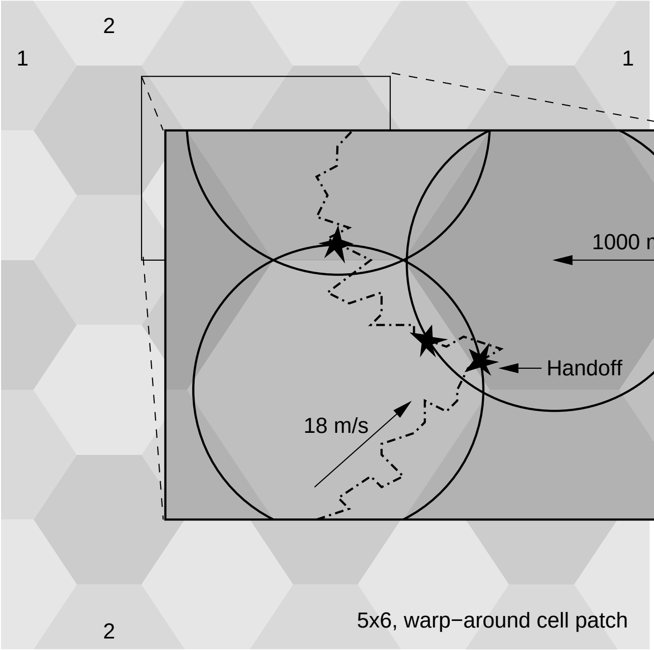


Figure 6.3: The simulation setup. This picture gives an overview of the cells' organization. The dashed line shows a simulated mobile motion.

6.4.1 Simulations parameters

The simulations presented here closely mimic those of [CB00]. Their setup is the following.

The terrain is composed of 30 circular cells laid-out as a bee's nest. The cells have a 1000m radius and overlap, thus, introducing a "handover hysteresis" (see figure 6.3). The mobility model has been chosen so as to approximate the motion of a mobile in a random direction[1]. Both this direction and the mobile speed is updated at each time step according to a gaussian distribution. This model is precisely described in [Tol99] (each distribution has a unitary variance in radian for the direction, in meters per second for the speed, the mean speed is 18 m/s and the randomness factor a has been fixed to 0.4; this yields a mean cell traveling time of 110s). Figure 6.3 shows how a typical path looks like.

Using several cells (instead of doing the experiments with only one) allows to use reasonable cell traveling time and channel holding time distributions, without relying on approximations such as those presented in [ZD97].

Each simulation involves 4000 mobiles and lasts 12 hours.

The calls' duration is exponentially distributed with a mean of 3 minutes. New calls arrive according to a Poisson process whose frequency is adapted so as to

[1] The mobility model used in [Tol99] is not described precisely enough to be implemented here; thus, another realistic one has been used.

reach a target cell load, defined as

$$\text{Cell load} = \frac{\text{Arrival rate to the cell x Average call duration}}{\text{Number of channels per cell}}$$

Each cell has 18 channels. Each simulation is done with a given cell load. Both PCR1 and PCR3 have been tested.

As stated before, the handover prediction model is abstracted using 3 parameters: δ (the trigger-to-handover duration), σ (δ's standard deviation) and π (the prediction success ratio); their effect is studied in the following. When a missed prediction occurs, the predicted cell is chosen randomly in the cells neighbouring the one where the mobile will eventually go. The trigger-to-handover duration is modelled by a gaussian distribution with mean δ and variance σ^2; the trigger is discarded if this duration is greater than the mobile's cell travelling time.

6.4.2 Simulations results

6.4.2.1 Handover blocking ratio

The result of the simulations is given in the graphs composing figures 6.4 and 6.5. A reservation is done as soon as a trigger is emitted; the reservation duration is thus equal to the trigger-to-handover delay. The new call blocking rate is an important factor that helps estimating a CAC scheme's efficiency; it is studied in the next section.

All the plots draw handover blocking percentages as a function of the successful prediction ratio and the trigger-to-handover delay (here always exactly equal to δ, that is $\sigma = 0$; section 6.5 studies the effect of σ). Prediction ratios as low as 0.2 have been studied; it is only a little bit higher than the worst prediction scheme, which would take the next cell at random among the neighbouring cells.

The zero δ borderline case shows what happens when no reservation takes place (since every reservation lasts a very short amount of time); this explains why all the lines merge when δ is small and gives a reference blocking ratio that should not be exceeded (since in this case introducing CAC is worse than doing nothing).

All the graphs related to the same algorithm share a common look. In all cases, a better prediction clearly gives a better blocking ratio in return. The results obtained with a 50% cell load are not very significant: the blocking rates are so low that they make reservations useless.

PCR1 performance are maximized when the reservations last about 5 seconds; long reservations can even give worse results than no reservation at all. As expected, when the prediction is more accurate, the reservations can last longer (up to 10s) since there are less likely to block a channel in a wrong cell. With this particular algorithm, the reservation duration is an important parameter that is discussed in section 6.5.

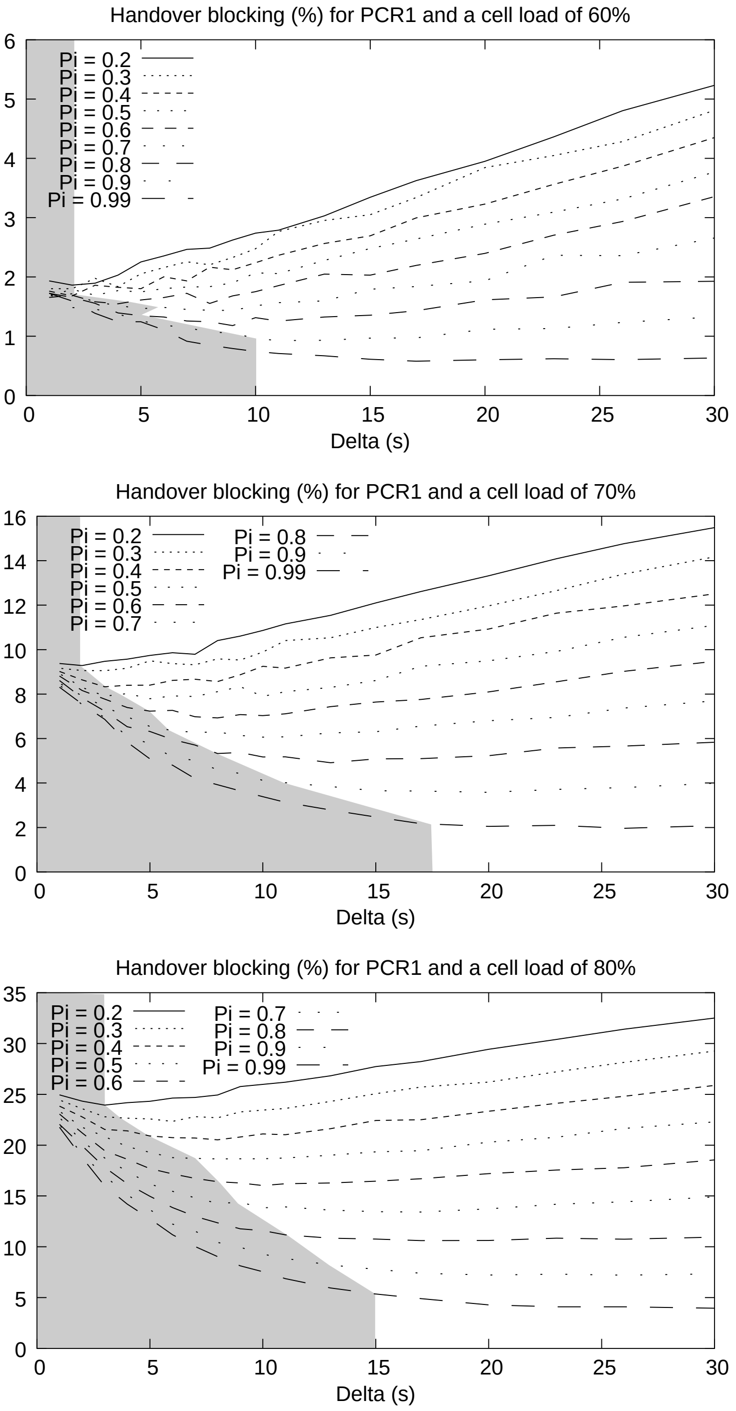


Figure 6.4: Handover blocking rate *versus* prediction parameters for PCR1. The meaning of the grey zones is explained in section 6.4.2.2.

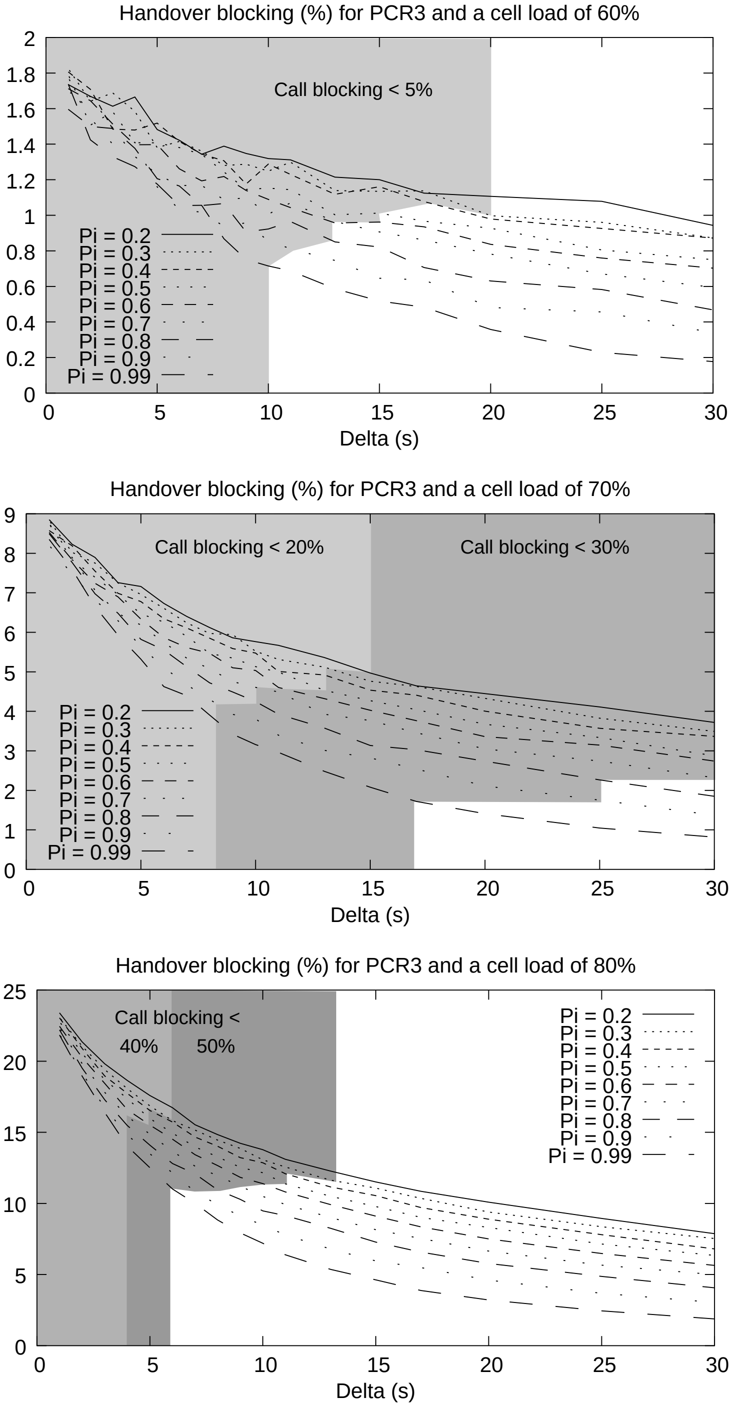


Figure 6.5: Handover blocking rate *versus* prediction parameters for PCR3. The cell loads are the same as those used in figure 6.4. The meaning of the grey zones is explained in section 6.4.2.2.

The plots related to PCR3 have a very different shape. For a given prediction ratio, the handover blocking curve decreases with the reservation duration; one can thus see the advantage brought by the fact that reservations are not associated with a particular mobile anymore.

6.4.2.2 New call blocking ratio

Even if it is preferable not to allow a new call rather than interrupting an ongoing one, one should not forget that there is a tradeoff between those two blocking ratios.

The decreasing section of a curve describing the behaviour of one ratio with respect to the other can be seen, in a sense, as the Pareto front of a multi-criteria optimization. All the front's points are Pareto-optimal and the choice of a specific one depends on the considered system.

Figure 6.6 shows a plot for a PCR1 cell loaded at 70% in the same conditions as those explained in section 6.4. All the points of the decreasing section of the curves are optimal: there is no strict rule to choose one of them.

One can see that as the next cell prediction becomes more accurate, reaching the curve's minimum requires higher a call blocking. A way to approach this minimum while keeping the call ratio between reasonable bounds for high values of π is to restrict the allowed value of the derivative. The figure's grey zone holds the points where reducing the handover blocking by one percent costs less than three percents

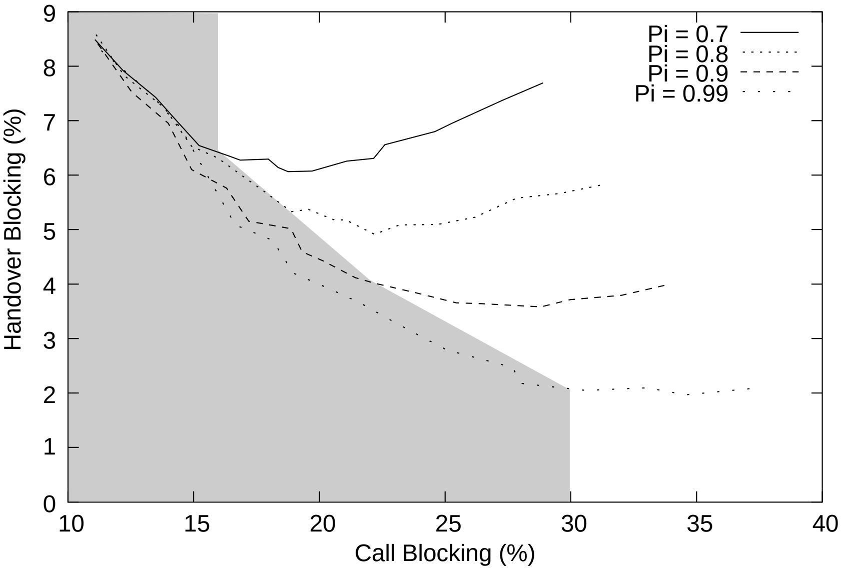


Figure 6.6: Handover *versus* call blocking rate in a cell loaded at 70%, using PCR1. The grey zone shows points where the absolute value of the derivative is higher than 1/3: gaining 1% in handover blocking costs less than 3% in call blocking. The points at the border of this zone could thus be considered as optimal.

in call blocking. This zone has also been drawn on figure 6.4.

In the curves depicted in figure 6.5, another strategy has been used. It has been chosen to color the plots according to the call blocking value. The curves' intersection points with those colored zones show how the handover blocking varies with respect to the next cell prediction accuracy for a constant call blocking value.

6.5 Timing

6.5.1 The ideal timing

One could try to take advantage of the local minimum shown in the plots related to PCR1 (figure 6.4).

In the context presented here, reserving T seconds on average amounts to wait $\delta - T$ seconds after each trigger, yet this suppose that the error on δ is symmetric and has a mean of zero. In what follows, we will suppose that the distribution of this error is gaussian and has a standard deviation given by σ.

This method can't of course be applied to prediction schemes that don't estimate a δ value associated with each trigger.

To verify the effectiveness of this idea, we can use simulations similar to those presented in section 6.4. Figure 6.7 shows how the handover blocking ratio changes with respect to the prediction ratio π and the prediction accuracy σ. In this figure, the mobiles reserve a channel during $\mathrm{N}(T, \sigma)$ seconds[2] before the handover; negative values are discarded (no prediction trigger is emitted). The other parameters are the same as those used in section 6.4, the cell load has been fixed to 70% and the chosen optimum is the minimum of the corresponding plot in figure 6.4 (and is thus a function of π).

One can verify that the values obtained when σ equals 0 are the same as those appearing in figure 6.4.

6.5.2 Tradeoff

The resulting plot shows the effect of a rough handover time estimation. As one could have guessed, the next cell prediction is the main goal in order to achieve good blocking ratios; however, figure 6.7 shows that this assumption is not always true for low values of π.

For example, to improve the blocking ratio by 1%, a prediction scheme characterized by (π, σ) parameters equal to $(.6, 5)$ can reach $(.6, 1)$ or $(.7, 5)$. For most schemes, this makes a serious difference and one of those points will be easier to

[2] $\mathrm{N}(T, \sigma)$ is a random variable from a gaussian distribution with a mean equal to T and a standard deviation equal to σ.

achieve than the other.

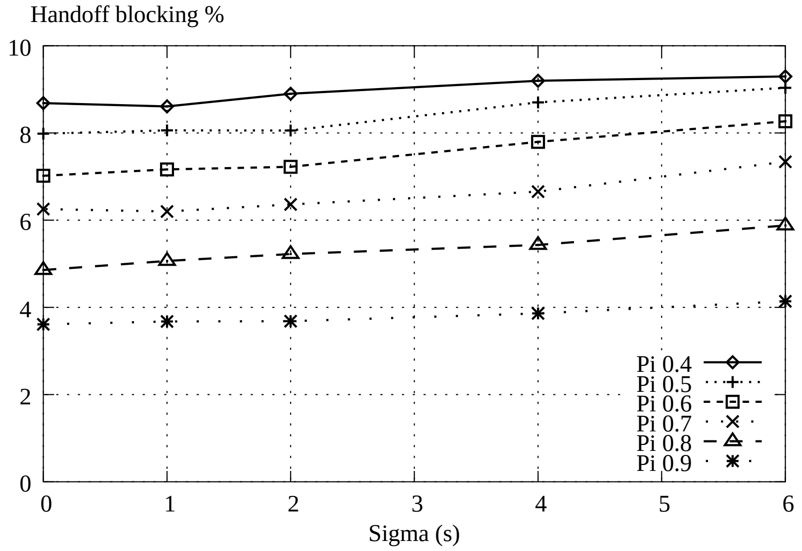


Figure 6.7: The effect of π and σ on PCR1 performance. This plot is similar to those given in figure 6.4, where the reservation delay is always 5 seconds plus a random, gaussian value characterized by a standard deviation of $\boldsymbol{\sigma}$ seconds.

6.6 Optimal duration as a function of network parameters

We have seen that PCR1 has an optimal reservation duration once the mobiles movement behaviour has been fixed. One could wonder how to choose a reservation duration given a certain wireless network's mobiles characteristics.

If one considers the mobiles' calling behaviour (*i.e.* the calls frequency and duration, which determine the cell load) and prediction parameters as fixed, the only way a mobile can modify the handover blocking probability is by changing its movement pattern. In this context, the mean duration between the handovers is certainly the most important factor.

Figure 6.8 thus plots the optimal reservation duration as a function of the mobiles' mean cell traveling time. The simulations that yielded those results are similar to those presented in section 6.4, but with various mobile speeds.

As expected, the optimal duration decreases with the traveling time. This is a direct consequence of the higher handover frequency: without shorter reservations, the reservation queue would exceed its optimal length.

It is not surprising to see that the various curves' slope grows together with the next cell prediction accuracy because *(a)* the higher the value of π, the longer

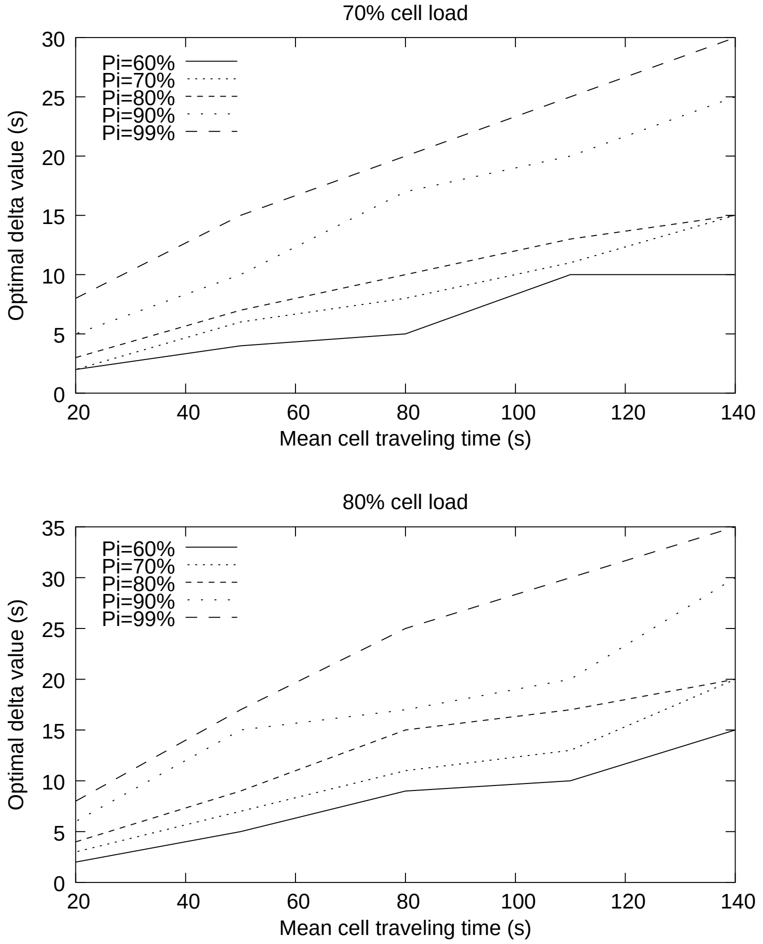


Figure 6.8: Optimal reservation duration as a function of the mean cell traveling time with different cell loads and next cell prediction ratios. The prediction parameter $\boldsymbol{\sigma}$ is equal to zero.

the optimal duration (for a fixed mobile speed) and *(b)* when the speed is high, durations are shorter and their absolute difference tends to get thinner.

6.7 Merit-based Admission Control

The results presented in the previous section lead to a natural idea: it would be desirable to modify the CAC scheme in order to favour mobiles which are trying to reserve channels at a proper time (*i.e.* minimizing the handover blocking of the whole system).

PCR1 can be modified as follows to achieve this goal; when a mobile m performs a handover:

1. If it has a reserved channel, it is allocated;
2. If there is a free channel, it is allocated;
3. If the mobile has a pending reservation (*i.e.* in the reservation queue), it applies a *fairness test* (discussed below) which tries to find a less meriting reservation r'. If it succeeds, the channel associated with r' is allocated to m and r' is pushed back at the beginning of the queue (r' will be the next reservation to be popped out of the queue since it was already associated with a channel).
4. Else, the handover is blocked.

Thus, the idea is that if a mobile m has a pending reservation when it performs its handover, it should be allowed to borrow the effective (*i.e.* associated with a channel) reservation of another mobile m' if the reservation behaviour of m is more likely to reduce the overall blocking ratio than m''s. A first, simple *fairness test* is thus one that favours mobiles near the optimal reservation duration T pointed out in section 6.5.1. Let t be the duration[3] of the pending reservation r of the mobile performing a handover and t' the duration of an effective reservation r' that could be borrowed.

Simple test

This test allows to borrow r' if $|T - t| < |T - t'|$; this simply means that a reservation can be borrowed if it is further apart the optimal time (*i.e.* reservations close to T will be left untouched).

The modified algorithm has three strong points.

First, it adds some fairness to the system; very long reservations are not profitable anymore, since the longer, the more likely they are to be replaced by a shorter, more sensible one thanks to the *fairness test.* One could say that this scheme favours mobiles meriting it, *i.e.* those that try to get a fair amount of the network's resources.

Second, since a reservation that lasts for about T time units is rewarded, the overall handover blocking ratio decreases.

Third, the modification has a minimal impact on the new call blocking ratio. Indeed, the above procedure's third item does not add any new reservation, but borrows a channel to allow a handover which would otherwise have been blocked. It could prevent a new call to take place, but allows a handover to occur instead: this is exactly the spirit of the admission control we want to define.

Figure 6.9 plots the results of a simulation of this algorithm. Mobiles have been divided into four classes with different mean reservation durations: 5, 10, 20

[3] The *duration* of a reservation is defined as the amount of time between the reservation creation and the mobile handover.

and 30 seconds (gaussian distributed with a standard deviation of 1 second, the distribution is thus a *gaussian mixture*); they are experiencing cell loads of 60, 70 and 80%. Each class has the same number of mobiles and all of them use a π factor equal to 0.7. Since the simulation parameters have been chosen equal to those taken in section 6.4, the ideal reservation duration T has been fixed to 8, 10 or 15 seconds (depending on the cell load and according to the results plotted in figure 6.4).

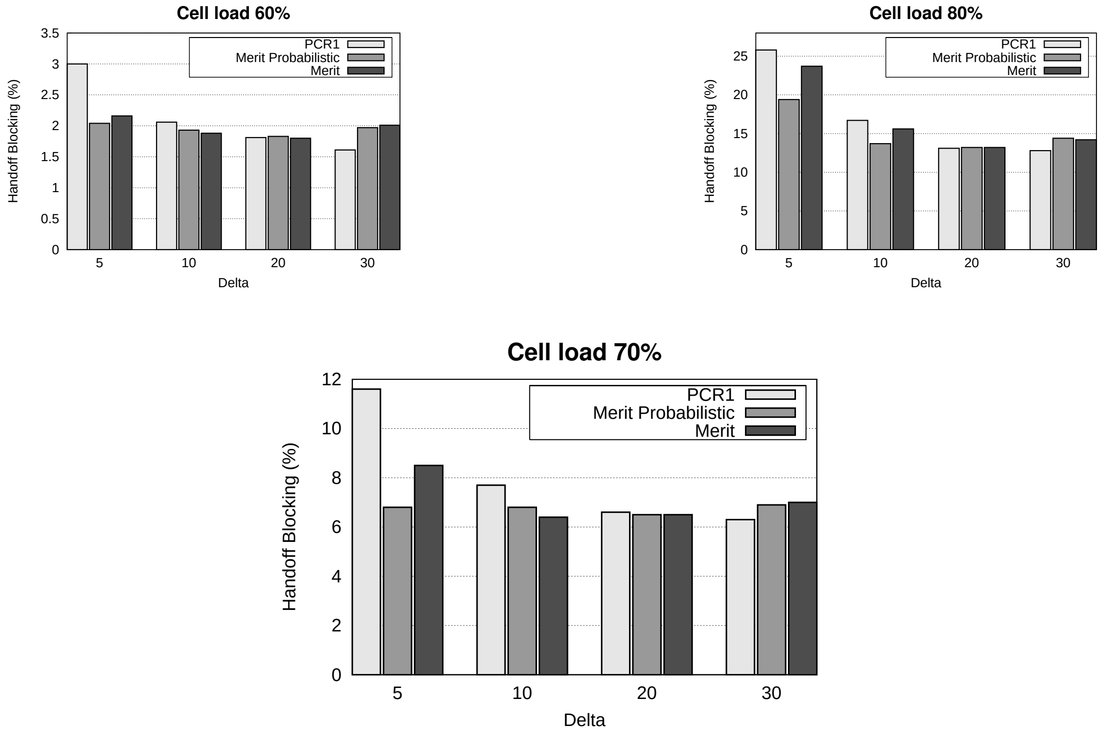


Figure 6.9: This simulation shows the handover blocking ratios experienced by four classes of mobiles under a 60, 70 and 80% cell load. They are characterized by different mean reservation durations (from 5 to 30 seconds), otherwise the simulation parameters are the same as those described in section 6.4. The chosen value of T are consistent with the results shown in figure 6.4.

This figure shows the results achieved by another *fairness test*:

Probabilistic test

Given the distribution of reservation durations d, one can compute the expected distance to T of r' knowing that it is at least t'. The test is thus $|T - t| < E[|x - T|]$, where x is the random variable associated with reservation durations (thus greater than t'):

$$E_{x \geq t'}[|x - T|] = \frac{\int_{t'}^{\infty} d(x)|x - T|\,dx}{\int_{t'}^{\infty} d(x)\,dx} \tag{6.1}$$

Notice that this requires the knowledge of the distribution d, which imposes monitoring mobiles' reservations durations.

	Cell load		
Method	**60%**	**70%**	**80%**
PCR1	2.1%	8%	17.1%
Merit (simple)	2.0% (7.8%)	7.1% (12.8%)	16.7% (2.3%)
Merit (probabilistic)	1.9% (9.2%)	6.7% (19.6%)	15.2% (12.6%)

Table 6.1: Handover blocking obtained with and without the different merit-based admission control mechanisms.

Both fairness tests have a common drawback: mobiles doing reservations at the right moment still could have their reservations borrowed; they only are favoured statistically.

Choosing a different anchor timing could still add fairness to the system; here, fairness and performance are orthogonal issues.

The simulation's results emphasize that PCR1 is not fair: mobiles reserving channels for a longer period of time experience a better service at the expense of the global handover blocking ratio. On the other hand, the results of merit-based methods are divided up homogeneously. In particular, classes characterized by low and high δ parameters systematically see their blocking decrease or increase, respectively.

The advantage of the more complicated, probabilistic test is revealed by table 6.1. It clearly shows that it performs better in terms of handover blocking.

The figures in parentheses give the relative difference with PCR1. The simple test doesn't take care of the fact that reservation lasting for about T time units are likely to last during a much longer amount of time. This behaviour is particularly harmful to mobile classes with small δ values.

As expected, the call blocking is not that much affected by the merit-based schemes; relatively to PCR1, the simple test increases it by about 2% while the probabilistic one increases it by about 4%.

The difference between the two merit tests is brought to the fore by figure 6.10. The first plot shows the reservation time estimated by the probabilistic (equation 6.1) and the simple test (which approximates $E_{x \geq t'}[|x - T|] \simeq |t' - T|$). Several reservation duration densities $d(\cdot)$ have been considered:

- the gaussian mixture used in the simulation above (*i.e.* 4 gaussian distributions centered on 5, 10, 20 and 30 seconds with an extended deviation of 1 second);

- uniform densities over $[0, M]$ with $M = 10$, 20 or 30 seconds.

The optimal time is T=10s.

This plot helps to see which effective reservations can be borrowed: a mobile

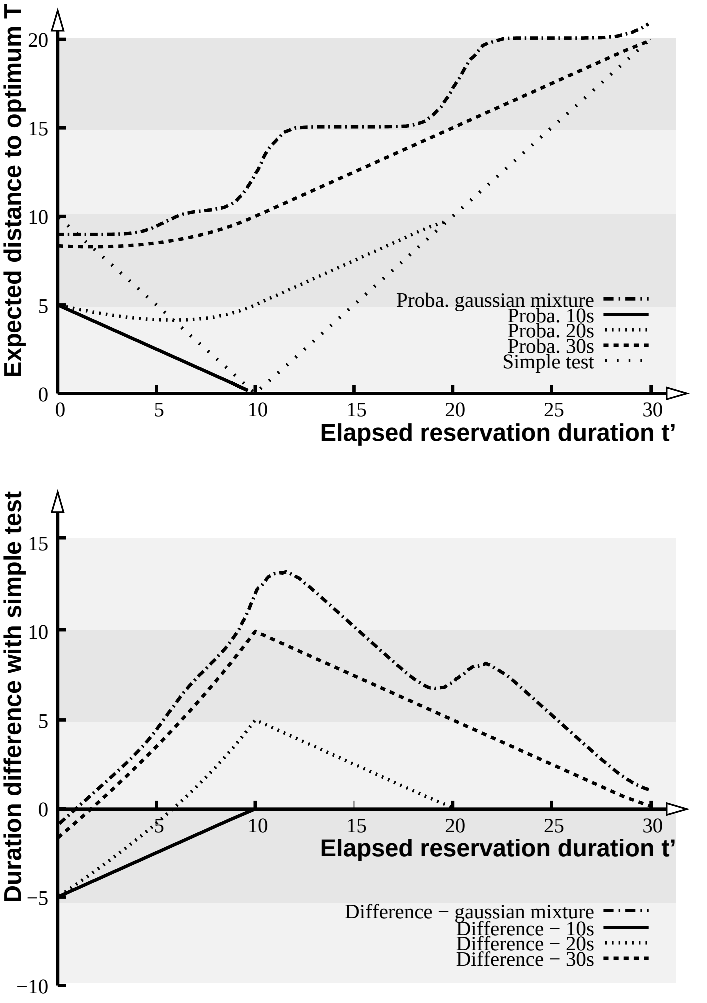


Figure 6.10: Comparision between the *probabilistic* and *simple* tests. The curves of the first plot draw the reservation duration estimations for two reservation duration distribution types; the first is the gaussian mixture used in the simulation above, the second a uniform distribution over an interval of 10, 20 or 30 seconds. The optimal time is T=10s.

that has a pending reservation that lasted t seconds can borrow an effective reservation that has already lasted t' seconds if the curve ordinate at abscissa t' is greater than $|t - T|$.

The second plot draws the difference between the curves relative to the probabilistic test and the curve relative to the simple test. The positive portion of the curves shows when borrowing an effective reservation is easier with the probabilistic test (*i.e.* borrowing is allowed for a longer interval centered on T).

The simple test is clearly over-pessimistic when most reservations last less than the optimal time, and over-pessimistic when dealing with long reservations. This fact explains the different behaviour of the merit tests. Considering the distribution related to the simulation above, one can observe that a mobile performing a handover with a pending reservation that lasted t seconds —with $\pm 1 < t < \pm 20$, *i.e.* a difference to the optimal time less than about 9s— can borrow any other effective reservation using the probabilistic estimation, but cannot borrow those that lasted between t and $20 - t$ seconds with the simple test. Being centered on the optimal time, the 10s class does not experience this difference (for those mobiles, the $[t, 2T - t]$ range is small). The 5s class is, on the contrary, badly off: its mobiles cannot borrow reservations in the 5 to 15 seconds range (this is important since most reservations last more than 5 seconds, and explains the high blocking ratio experienced by this class in the simulations). Longer reservations experience less problems, since the probabilistic curve come closer to the simple test's and less mobiles are likely to be involved in long reservations.

6.8 Conclusion

It is usually admitted that motion prediction can enhance the quality of service provided by wireless networks. This chapter has discussed how the accuracy of the prediction acts on the level of service in the particular case of cell access control.

It has been shown that, with certain access control protocols, reservations must be done at the proper time; this ensures that the handover blocking ratio is minimized and that channels are not reserved for too long. Being able to perform a reservation at the proper time implies some conditions on the prediction mechanism, which have been quantified using simulations based on an abstract prediction model.

This naturally leads to a modified access control scheme that penalizes early and late reservations. This has two consequences: the network fairness is improved (selfish mobiles are discouraged) and the overall handover blocking is minimized. This has been confirmed by simulations.

In the future, the impact of movement prediction's accuracy on other protocols should be studied. The analysis of existing prediction schemes could allow one to setup a more accurate abstract model than the one presented here.

7

Routing a Disruption Tolerant Network

He who refuses to do arithmetic is doomed to talk nonsense.

— JOHN McCARTHY, *Progress and its Sustainability* (1995)

THE research community has recently expressed a strong interest in *Disruption Tolerant Networks* — *i.e.* ad hoc networks which can temporally be disconnected. This type of network models quite accurately the swarm of short-distance wireless devices we commonly own, be it a GSM or a PDA.

Since the (dynamic) topology of such networks is directly determined by the motion of the mobiles it is made of, and since users do not move randomly, it is expected that the network topology can be guessed in advance. This knowledge of the future network topology could be exploited by routing algorithms.

This chapter defines a framework that describes mathematically the predictability of the future mobile nodes' contacts. Using this framework, the expected end-to-end packet delay can be computed once the packet route has been determined.

As an example of how the framework can be used, we demonstrate that it allows one to elegantly adapt the Bellman-Ford algorithm to predictable Disruption Tolerant Networks.

7.1 Introduction

Disruption (or *Delay*) *Tolerant Networks* (DTNs, [Zha06]) have been the subject of much research activity in the last few years, pushing further the concept of ad hoc networks. Like ad hoc networks, DTNs are infrastructureless, thus the packets are relayed from one node to the next until they reach their destination. However, in DTNs, node clusters can be completely disconnected from the rest of the network. In this case, nodes must buffer the packets and wait until node mobility changes the network's topology, allowing the packets to be finally delivered.

A network of Bluetooth-enabled PDAs, a village intermittently connected *via* low Earth orbiting satellites, or even an interplanetary Internet ([BHT$^+$03]) are examples of disruption tolerant networks.

The atomic data unit is a group of packets to be delivered together. In DTN parlance, it is called a *message* or a *bundle*; we use the latter in the following.

Routing in such networks is particularly challenging since it requires to take into account the uncertainty of mobiles movements. The first method that has been proposed in the literature is pretty radical and proposes to forward bundles in an "epidemic" way ([VB00, SH03, JOW$^+$02]), *i.e.*, to copy them each time a new node is encountered. This method of course results in optimum delays and delivery probabilities, at the expense of an extremely high consumption of bandwidth (and, thus, energy) and memory. To mitigate those shortcomings, the epidemic routing has been enhanced using heuristics that allow the propagation of bundles to a subset of all the nodes ([TR01, SPR04, SPR05]).

Since node's buffer memory is not unlimited, a cache mechanism has been proposed, where the most interesting bundles are kept (*i.e.* those that are likely to reach their destination soon) and the others are discarded when the cache is full ([NPB03, WJMF05, LDS03, TZZ03, WJMF05, JLW05, LFC05]). Those schemes must thus guess when a bundle will reach its destination, which is most of the time computed thanks to frequency contact estimation (which reflects the probability that two given nodes meet in the future). We examine such a scheme in more details in the next section.

Few papers explore how the expected delay could be more precisely estimated (notable exceptions are [SBRJ02, MHM05]). It has been proved ([MAZ04]) that a perfect knowledge of the future node meetings allows the computation of an optimal bundle routing.

This short overview emphasizes two shortcomings:

- Certain networks might be highly predictable (*e.g.* nodes are satellites and links appear and vanish as they revolve around their planet), others are much more chaotic. Previous work suppose either that nodes contacts are perfectly deterministic and known in advance, or that only the contact frequency is known for each pair of nodes. We propose to generalise these approaches and suppose that each node knows a probability distribution of contacts in the (near) future.

- [JOW$^+$02] underlines the tradeoff between bundle delivery guarantees and bandwidth/energy consumption: copying the bundles is costly since, in mobile networks, those resources are scarce. Current schemes use a cache mechanism that ensures each node only receives the most relevant bundles, which somehow mitigates this problem, but does not provide any rationale, except the need to cope with mobiles limited memory. We propose to adapt the bundle routing according to the delivery or delay guarantees required by the user.

We have defined a framework that allows one to formalize the behaviour of a predictable DTN. It defines precisely how the mobiles motion predictability can be modeled and, given a routing algorithm, uses this predictability to estimate at what pace the bundle reaches its destination.

As an example of how this framework can be used, we give a packet forwarding algorithm optimized to meet the delay/bandwidth consumption trade-off: packets are duplicated so as to (statistically) guarantee a given delay or delivery probability, but not too much so as to reduce bandwidth, energy, and memory consumption.

7.2 Previous work

In this section, we briefly explain the existing DTN routing protocol we find the most similar to the one proposed in this thesis ([Zha06, BBL05]) . This should help the reader to better grasp the point of view we try to emphasize.

As in our work, this method tries to estimate the end-to-end delivery probability and delay, and uses this estimation to route bundles efficiently.

The method assumes that the bundles are destined to a given stationary location (*e.g.* a given office). Bundle transmission delays are neglected, but we consider that each node has limited bundle buffering capabilities.

Assuming that a node's probability of visiting a location is strongly correlated with its past visit history, the probability that node k visits location i — denoted $P_0^k(i)$ — is given by:

$$P_0^k(i) = t_i^k / t$$

where t_i^k is the number of rounds node k visited location i in the past t rounds.

We can now compute the probability that the bundle is successfully delivered to location i, starting with node k and with the help of at most one intermediate node. We have:

$$P_1^k(i) = 1 - \prod_{j \text{ in node set}} \left(1 - m_{j,k} P_0^j(i)\right)$$

where $m_{j,k}$ is the meeting probability of nodes j and k, estimated thanks to the past meeting rate ($m_{k,k} = 1$).

We can finally generalize to n hops (at most):

$$P_n^k(i) = 1 - \prod_{j \text{ in node set}} \left(1 - m_{j,k} P_{n-1}^j(i)\right)$$

When nodes j and k meet, they compare their list of to be delivered bundles. Bundles are exchanged between j and k using an empirical algorithm (details can be found in [BBL05]); this exchange is such that:

- The mobiles' buffer capacity is never exceeded.
- Bundles are exchanged if the other peer can more effectively carry it to its destination i. This is estimated by comparing $P_n^j(i)$ and $P_n^k(i)$ (with n sufficiently high; [BBL05] claims that low values of n already give a good estimation of the end-to-end delivery probability).

We can observe that this method tries to extrapolate how frequently nodes will meet in the future, but does so very roughly. We propose a more nuanced scheme which uses as much information as possible from the knowledge of the future nodes movements.

7.3 Predictable future contacts

The network is composed of a finite set of wireless nodes $\mathcal{N}$ that can move and thus, from time to time, come into contact.

In the sequel, a *contact* between two nodes happens when those nodes have setup a bi-directional wireless link between them. A contact is always considered long enough to allow all the required data exchanges to take place[1].

7.3.1 Contact profiles

We expect the mobiles motion to be, to a certain extent, predictable, yet obviously the degree of predictability varies from one network to another. Sometimes nodes motion is known in advance because they must stick to a given schedule (*e.g.* a network of buses) or because their trajectory can easily be modelled (*e.g.* nodes embedded in a satellite). Other networks are less predictable, yet not totally random: colleagues could be pretty sure to meet every day during working hours, without any other time guarantee. Mobile nodes behaviour could also be learnt automatically so as to extract cyclical contact patterns.

We therefore suppose that each node pair $\{a, b\} \subset \mathcal{N}$ can estimate its contact probability for each (discrete) time step in the near future. We call it a *contact profile* and denote it $C_{ab} : \mathbb{N} \to [0, 1]$. The time step duration should be chosen small compared to the expected network's end-to-end delay. Figure 7.1 gives an hypothetical contact profile. In the following, we suppose the profile known for each node pair.

Contact profiles can easily represent situations usually depicted in the literature:

- A constant profile $C_{ab}(t) = k$ describes a node pair that only knows its contact frequency. For example, the profile $C_{ab}(t) = 1/30$ (probability of contact per day) corresponds to two nodes a and b meeting once a month on average.

[1] This is a major difference with [MAZ04] which does not neglect bundle transmission times.

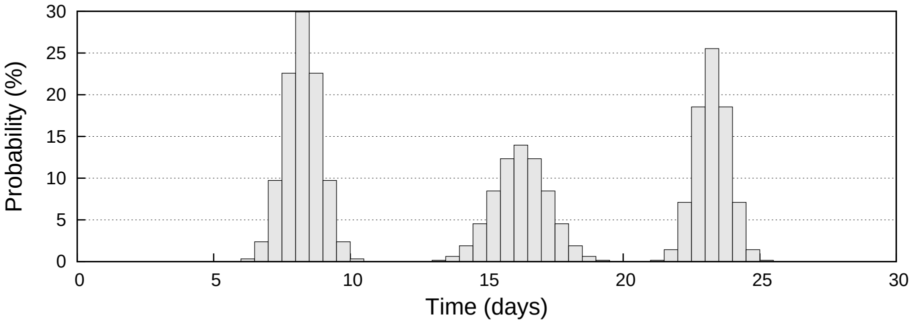


Figure 7.1: Example of contact profile of a node pair over a month. The height of a bar gives the probability that two nodes meet (*at least* once) during the corresponding 12-hour time period. Here, nodes are supposed to meet at the beginning of each week, but the exact day is unknown.

- Perfect knowledge of nodes meeting times results in a profile made of peaks: $\forall t \in \mathbb{N} : C_{ab}(t) \in \{0, 1\}$.

In practice, unknown contact profiles could be replaced by a constant function equal to zero on its domain to get a defensive approximation of their behaviour.

The following sections aim at studying how bundles propagate from one node to another in a network whose nodes' contact profiles are known.

7.3.2 First contact distribution

It is easy to deduce the probability distribution of a (first) contact at time t between nodes a and $b \in \mathcal{N}$ given their profile C_{ab}; we denote this distribution d_{ab}. Since the probability of a first contact at time t is the probability of meeting at time step t times the probability not to meet at time steps $0, 1, \ldots, t-1$, we have:

$$d_{ab}(t) = C_{ab}(t) \prod_{i=0}^{t-1} \bigl(1 - C_{ab}(i)\bigr) \qquad \forall a, b \in \mathcal{N},\ \forall t \in \mathbb{N} \tag{7.1}$$

The distributions domain is $\mathbb{N}$ since contact profiles have been defined using discrete time steps. We extend the distributions to $\mathbb{R}$ to get rid of this artifact. Notice that d_{ab} is not a well-defined probability distribution since its integral over its domain is not equal to 1: two nodes might never meet. Those considerations directly lead to the definition of the first contact distribution set.

Definition 7.1. *The first contact distribution set, $\mathcal{C}$, is the set of functions*[2] $f : \mathbb{R}^+ \to \mathbb{R}^+$ *such that* $\int_0^\infty f(x)\, dx \leq 1$.

[2] $\mathbb{R}^+$ denotes the set of positive reals.

Contact profiles have a shortcoming: they do not allow us to express contact interdependencies; for example, they cannot model that two nodes are certain to meet exactly one during the weekend without knowing exactly which day (if a probability of .5 is assignated to Saturday and Sunday, there is a .25 probability that the nodes will meet twice). First contact distributions have no such limitations. Therefore, when it is possible, one could find preferable to generate them directly without relying on contact profiles.

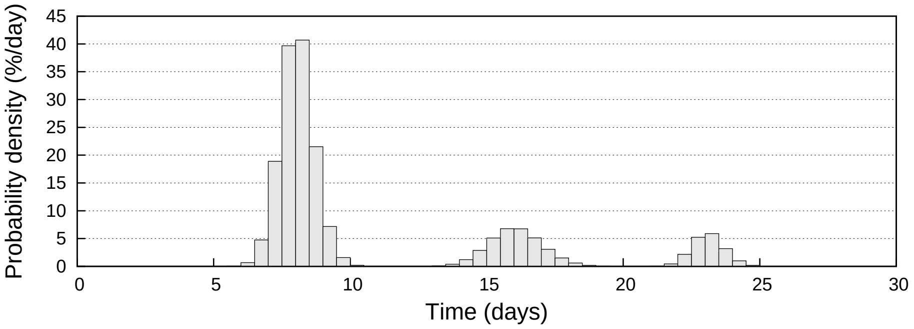


Figure 7.2: First contact probability distribution corresponding to the contact profile figure 7.1. (Each bar corresponds to a 12-hour period.)

Figure 7.2 gives the d_{ab} distribution corresponding to the contact profile C_{ab} depicted in figure 7.1.

Notice that if a bundle is delivered directly from a to b, knowing the first contact distribution allows an easy verification of a large spectrum of guarantees, such as the average delay or the probability of delivery before a certain date.

7.4 Delivery distributions

7.4.1 Definition

First contact distributions can be generalized to take into account the knowledge that no contact were made before a certain date.

Let $D_{ab}(T,t)$ be the probability distribution that a and b require a delay of t time steps to meet for the first time after time step T. Since these distributions will be the building blocks that allow us to compute when a bundle can be delivered to its destination, we call them *delivery distributions*. D_{ab} can directly be derived from the contact profile C_{ab}:

$$D_{ab}(T,t) = C_{ab}(T+t) \prod_{i=T}^{T+t-1} \big(1 - C_{ab}(i)\big) \qquad \forall a,b \in \mathcal{N},\ \forall T,t \in \mathbb{N} \tag{7.2}$$

As before, the domain of these functions can be extended to ${\mathbb{R}^+}^2$.

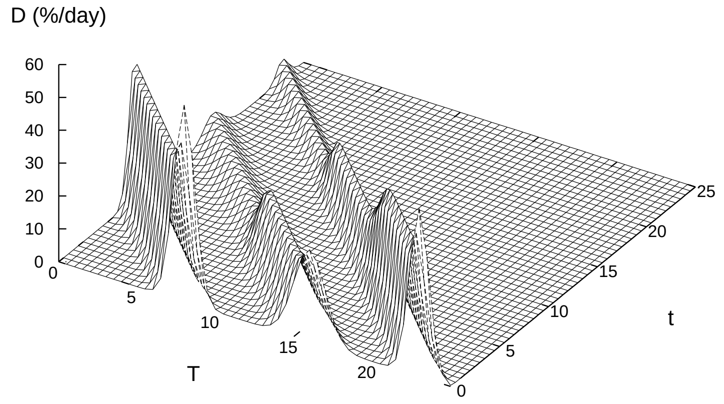


Figure 7.3: The $D_{ab}(T, t)$ function matching the contact profile given in figure 7.1.

Definition 7.2. *The* delivery distribution set*, $\mathcal{D}$, holds all the functions $f : \mathbb{R}^{+2} \to \mathbb{R}^{+}$ such that $\forall T \in \mathbb{R}^{+} : \int_0^\infty f(T, x)\, dx \leq 1$.*

Notice the inequality.

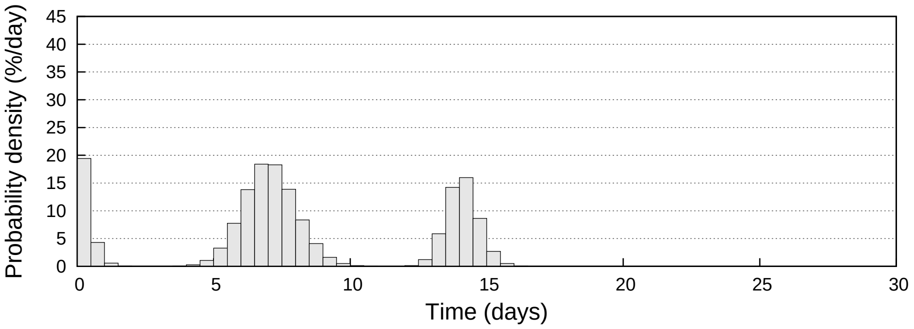


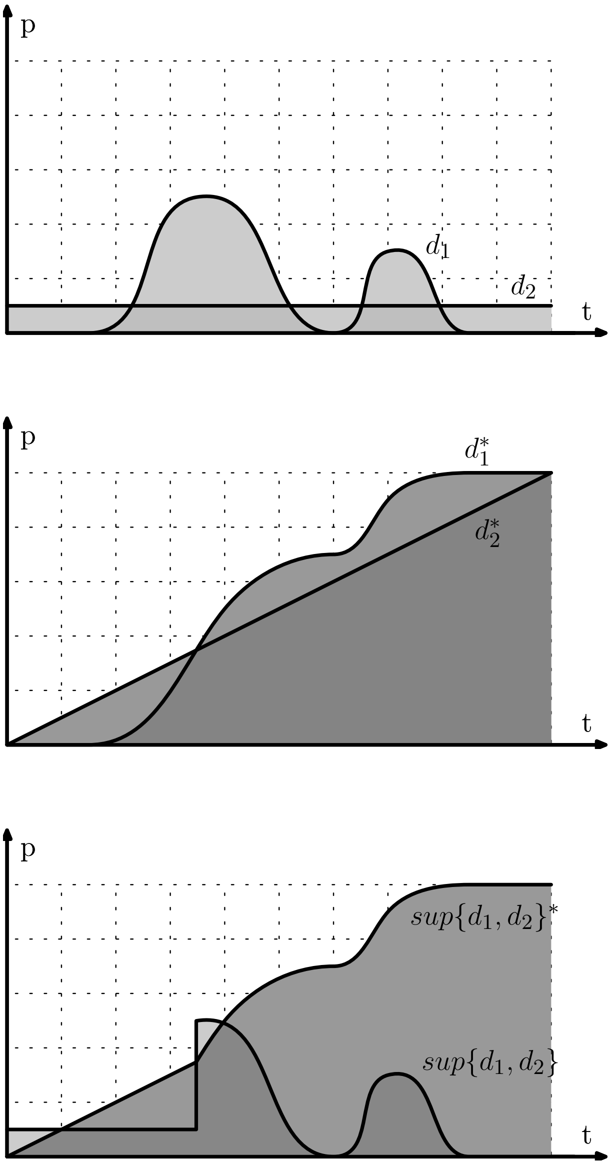
Figure 7.4: The contact probability density $D_{ab}(9, \cdot)$ matching the contact profile given in figure 7.1.

The $D_{ab}(T, t)$ distribution corresponding to the contact profile given in figure 7.1 is plotted in figure 7.3. Figure 7.4 plots the function $D_{ab}(9, \cdot)$ (*i.e.* a section of $D_{ab}(T, t)$ in the $T = 9$ plane); the $D(T, \cdot)$ functions of course belong to $\mathcal{C}$ ($\forall T \geq 0$).

Notice that $D_{ab}(T, \cdot)$ is the expected delivery delay distribution for a bundle sent directly from a source a to a destination b if a decides to send it at time T.

7.4.2 Order relation on distributions

We define an order relation between first contact distributions. Intuitively, this relation allows one to compare two distributions to find which one represents more frequent or predictable contacts. A rigorous definition is given below.



Definition 7.3 specifies when two contact distributions d_1, d_2 are such that $d_1 \succeq d_2$. The plots show two distribution examples (top) and their cumulative function (denoted d_1^* and d_2^*, middle plot). We have $d_1 \succeq d_2$ iff $\forall t \geq 0 : d_1^*(t) \geq d_2^*(t)$. Here, neither $d_1 \succeq d_2$ nor $d_2 \succeq d_1$ hold.

The distribution $sup\{d_1, d_2\}$ (bottom) is called *supremum* (or *least upper bound*). Its cumulative function is the maximum of the d_1^* and d_2^* functions; its distribution is the derivative of the cumulative function.

By definition of $sup\{d_1, d_2\}$, if $d \succeq d_1$ and $d \succeq d_2$, then $d \succeq sup\{d_1, d_2\}$ ($\forall d \in \mathcal{C}$). The infimum (or *greatest lower bound*) is defined in a similar manner.

Since every element of $\mathcal{C}^2$ has a corresponding supremum and infimum, the $\succeq$ relation defines a *lattice* structure on $\mathcal{C}$ (and on $\mathcal{D}$).

Figure 7.5: The $\succeq$ relation: example.

Definition 7.3. *The first contact distributions $d_1 \in \mathcal{C}$ is* greater (or equal) *than $d_2 \in \mathcal{C}$ (denoted $d_1 \succeq d_2$) if and only if:*

$$\forall x \geq 0 : \quad \int_0^x d_1(t)\,dt \geq \int_0^x d_2(t)\,dt \tag{7.3}$$

This relation is a *partial* order (but not a total order as there exist $d_1, d_2 \in \mathcal{C}$ such that neither $d_1 \succeq d_2$ nor $d_1 \preceq d_2$). Figure 7.5 gives an example of incomparable first contact distributions.

It appears difficult to define a total order on $\mathcal{C}$: comparing the distributions d_1 and d_2 in figure 7.5 is a matter of choice and depends on the bundle delivery guarantees one wants to enforce. The $\succeq$ relation is thus a least common denominator, and could be replaced in what follows with a more restrictive order definition.

The worst (smallest) element of $\mathcal{C}$ is the $\bot$ (*bottom*) distribution: $\bot(t) = 0$ $(\forall t \geq 0)$. The best (greatest) first contact distribution is denoted $\top$ (*top*): $\top(t) = \delta(t)$ $(\forall t \geq 0)$; the δ symbol denotes the Dirac distribution.

The $\succeq$ relation can be extended to $\mathcal{D}$. For all $D_1, D_2 \in \mathcal{D}$:

$$D_1 \succeq D_2 \iff \forall T \geq 0 : \; D_1(T, \cdot) \succeq D_2(T, \cdot)$$

The $D_\bot$ delivery distribution is such that $\forall T \geq 0 : D_\bot(T, \cdot) \equiv \bot$. The definition of $D_\top$ follows immediately.

7.5 Delivery distribution operators

7.5.1 The *forwarding* operator

Let D_{sbd} be the delivery distribution associated with the delivery of a bundle from a source node s to a destination d via node b. More precisely, if s decides to send a bundle at time T, it will reach d after a delay described by the $D_{sbd}(T, \cdot)$ distribution. D_{sbd} can be computed thanks to D_{sb} and D_{bd}:

$$D_{sbd} \equiv D_{sb} \otimes D_{bd} \tag{7.4}$$

The $\otimes$ (or *forwarding*) operator is a function defined for all distribution pair. We have $\otimes : \mathcal{D}^2 \to \mathcal{D}$:

$$\big(D_1 \otimes D_2\big)(T, t) = \int_0^t D_1(T, x)\, D_2(T + x, t - x)\, dx \tag{7.5}$$

It is easy to see that this operator is associative but not commutative.

Equation (7.5) simply states that since the total delivery delay is equal to t, if the delay to reach b is equal to x, then the delay from b to d is $t - x$.

Equation (7.4) can be generalized: a bundle could be forwarded through several intermediate hops before reaching its destination. We denote D_{s-d} (notice the dash) the delivery delay distribution for a bundle sent from a source s to a destination d at time T; from now on, $\otimes$ will thus be applied to any kind of delivery distributions.

For example, the graph below depicts a simple *delivery path*, *i.e.* a sequence of forwarding nodes; the corresponding delivery distribution is also given.

$$s \longrightarrow a \longrightarrow b \longrightarrow d \;: \quad D_{s-d} \equiv D_{sa} \otimes D_{ab} \otimes D_{bd}$$

We say that two delivery paths with a common source s and destination d are *disjoint* if the intersection of the set of nodes they involve is $\{s, d\}$.

7.5.2 The *duplication* operator

Let $D_{\substack{s-d\\ \searrow d}}$ be the delivery distribution associated with the delivery of a bundle from s to d if it is duplicated so as to follow the disjoint delivery paths described by the distributions D_{s-d} and D'_{s-d}. We have:

$$D_{\substack{s-d\\ \searrow d}} \equiv D_{s-d} \oplus D'_{s-d} \tag{7.6}$$

The $\oplus$ (or *duplication*) operator is a function $\oplus : \mathcal{D}^2 \to \mathcal{D}$, defined as follows:

$$\begin{split}\big(D_1 \oplus D_2\big)(T,t) = \left(1 - \int_0^t D_1(T,x)\,dx\right) D_2(T,t) + \\ \left(1 - \int_0^t D_2(T,x)\,dx\right) D_1(T,t)\end{split} \tag{7.7}$$

The expected delay computed is that of *the first* bundle to reach the destination d. It is easy to see that $\oplus$ is associative and commutative. Operators $\otimes$ and $\oplus$ can be combined to consider more complex forwarding strategies, assigning a higher precedence to $\otimes$.

Equation (7.7) is the sum of two terms. Each term is the probability that the bundle reaches the destination after a delay t using one path and that the bundle following the other path is not arrived yet.

Notice that we have both $D_1 \oplus D_2 \succeq D_1$ and $D_1 \oplus D_2 \succeq D_2$ (appendix, corollary C.1). This means that, contrary to what happens in deterministic networks, duplicating a bundle to send it along two paths can improve performance: it is not the case that the best path always delivers the bundle first.

The definition of this operator allows us to apply it to arbitrary independent distributions (for example, involving duplication and forwarding). This allows the computation of the distribution associated with a non trivial way to deliver a bundle, such as the one depicted below; the corresponding distribution formula is given on the right. Two arrows leaving a node depict a duplication.

$$\begin{tikzcd}
 & & e \arrow[r] & d \\
s \arrow[r] \arrow[dr] & b \arrow[r] \arrow[ur] & f \arrow[r] & d \\
 & c \arrow[rr] & & d
\end{tikzcd}
\quad : \quad
\begin{split} D_{s-d} \equiv \big(D_{sc} \otimes D_{cd}\big) \oplus \\ \big(D_{sb} \otimes (D_{be} \otimes D_{ed} \oplus D_{bf} \otimes D_{fd})\big)\end{split}$$

Figure 7.6 shows examples of the distributions obtained using those operators. As expected, the "duplication" operator shortens the delays and increases the delivery probability.

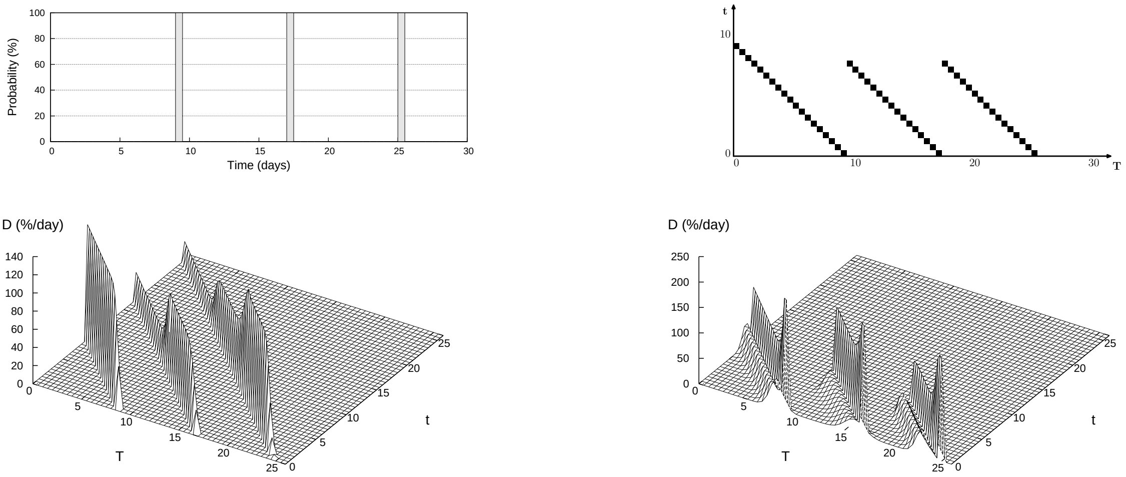


Figure 7.6: An example of the *forwarding* ($\otimes$) and *duplication* ($\oplus$) operators. We denote D_1 the delivery distribution depicted in figure 7.3. The top part of this figure depicts a contact profile (left) and the associated delivery distribution D_2 (dark squares represent a probability equal to 1). The left 3D plot depicts $D_1 \otimes D_2$, the right one $D_1 \oplus D_2$.

7.5.3 The *scheduling* operator

Let $D_{\substack{s-d \\ \ddots\, d}}$ be the delivery distribution that, every time a bundle has to be sent, chooses the best delivery strategy out of D_{s-d} and D'_{s-d}. We have:

$$D_{\substack{s-d \\ \ddots\, d}} \equiv D_{s-d} \oslash D'_{s-d} \tag{7.8}$$

The definition of $\oslash$ is straightforward. It is a function $\oslash : \mathcal{D}^2 \to \mathcal{D}$ such that:

$$\big(D_1 \oslash D_2\big)(T,t) = \begin{cases} D_1(T,t) & \text{if } D_2(T,\cdot) \not\succeq D_1(T,\cdot) \\ D_2(T,t) & \text{otherwise} \end{cases} \tag{7.9}$$

If s sends a bundle at time T, it is delivered using $D_2(T,\cdot)$ if and only if $D_2(T,\cdot) \succeq D_1(T,\cdot)$. This operator is not commutative since $\succeq$ is not a total order: when $D_1(T,\cdot)$ and $D_2(T,\cdot)$ cannot be compared, $D_1(T,\cdot)$ is chosen. This new operator can be combined with the other two ($\otimes$ and $\oplus$) assigning it a lower precedence.

The following example involves all the operators defined above. Two arrows leaving a node, one of them dotted, depict a scheduling operation. The dotted arrow leads to the second argument of $\oslash$, emphasizing the operator's non-commutativity.

$$\begin{tikzcd}
& & e \arrow[r] & d \\
s \arrow[r] \arrow[dr] & b \arrow[r] \arrow[ur, dotted] & f \arrow[r] & d \\
& c \arrow[rr] & & d
\end{tikzcd}
\quad : \quad
\begin{aligned} D_{s-d} \equiv \big(D_{sc} \otimes D_{cd}\big) \oplus \\ \big(D_{sb} \otimes (D_{bf} \otimes D_{fd} \oslash D_{be} \otimes D_{ed})\big) \end{aligned}$$

7.5.4 Delivery schemes

We have defined a *delivery path* as a delivery strategy that only involves forwarding.

A *delivery scheme* with source s and destination d is a general delivery strategy that allows a bundle to be delivered from s to d. It can use an arbitrary number of forwarding, duplication and scheduling operations. A delivery path is thus a particular delivery scheme.

Two delivery schemes from s to d are *disjoint* if the intersection of the set of nodes they involve is $\{s, d\}$.

7.6 Delivery guarantees

Knowing the delay distribution $d_{s-d} \in \mathcal{C}$ associated with the delivery of a bundle allows us to verify a large range of conditions on permissible delays or on delivery probabilities.

For example, the condition

$$\int_0^\infty d_{s-d}(t)\, t\, dt \leq d_{\max}$$

imposes a maximum expected delay $d_{\max}$, while

$$\int_0^{1\text{h}} d_{s-d}(t)\, dt \geq .9 \quad \text{and} \quad \int_0^{24\text{h}} d_{s-d}(t)\, dt \geq .99$$

matches distributions delivering a bundle in less than one hour nine times out of ten, and in less than a day with a probability of 99%.

We naturally impose that a condition fulfilled for a certain delivery scheme must be fulfilled for better schemes.

Definition 7.4. *A* delivery condition *C is a predicate: $C : \mathcal{C} \to \{\mathit{true}, \mathit{false}\}$ iff $\forall d_1, d_2 \in \mathcal{C}$ such that $d_1 \succeq d_2$, we have $C(d_2) \Longrightarrow C(d_1)$.*

A condition C can be extended to a delivery distribution $D \in \mathcal{D}$:

$$C(D) \iff \forall T \geq 0 : C\big(D(T, \cdot)\big)$$

7.7 Delivering bundles with guarantees

7.7.1 Probabilistic Bellman-Ford

Algorithm 7.1 adapts the Bellman-Ford algorithm to predictable disruption tolerant networks. In this section, *we do not allow bundle duplication.* Notice that, in

Algorithm 7.1: Probabilistic Bellman-Ford

```
   Data: $d$ is the destination node
 1 $\forall\, x \in \mathcal{N} \setminus \{d\} :\ B_x \leftarrow D_\perp$;
 2 $B_d \leftarrow D_\top$;
 3 repeat
 4     stabilized $\leftarrow$ true;
 5     forall $x \in \mathcal{N}$ do
 6         forall $y \in \mathcal{N}$ do
 7             $D_{xy-d} \leftarrow D_{xy} \otimes B_y$;
 8             if $B_x \neq B_x \oslash D_{xy-d}$ then
 9                 stabilized $\leftarrow$ false;
10                 $B_x \leftarrow B_x \oslash D_{xy-d}$;
11             end
12         end
13     end
14 until stabilized ;
```

general, the concept of "shortest path" is meaningless since the $\preceq$ relation is a *partial* order.

Similarly to the Bellman-Ford algorithm, algorithm 7.1 computes, for every node $n \in \mathcal{N}$, the best distribution leading to the destination found so far (B_n). This distribution is propagated to its neighbours (*i.e.* all the other nodes since the network is infrastructureless).

Once node x receives the best delivery distribution B_y found by y, it computes the delivery distribution obtained if it would send the bundle directly to y, and if y would forward it according to B_y. The resulting distribution is denoted D_{xy-d} (line 6).

D_{xy-d} is compared to the best known distribution to the destination (B_x) by means of the $\oslash$ operator. If D_{xy-d} is better than B_x on some time intervals, B_x is updated (line 9).

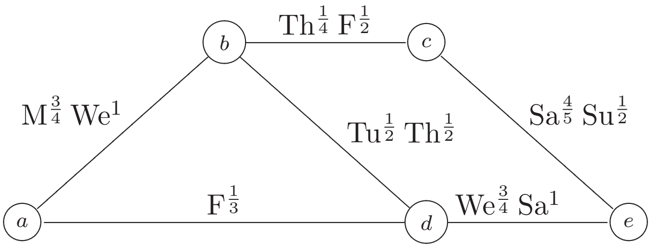
The algorithm terminates once no more B_x distribution is updated.

Figures 7.7 and 7.8 demonstrate how the algorithm works by means of a small example.

As mentioned before, this algorithm generalizes both [TZZ03] (*i.e.* converges to the *"shortest expected path"*) and [MAZ04][3] (*i.e.* finds the exact shortest path in the case of perfectly predictable networks).

[3] To be fair, this work also deals with message transmission delays, which are not considered here.

The delivery computed by this algorithm depends on the order at which the elements of $\mathcal{N}$ are picked up (lines 5 and 6). In practice, it might be preferable to rely on a heuristic to choose the preferred elements first.



b
c
a
d
e
$\mathrm{Th}^{\frac{1}{4}}\ \mathrm{F}^{\frac{1}{2}}$
$\mathrm{M}^{\frac{3}{4}}\ \mathrm{We}^{1}$
$\mathrm{Tu}^{\frac{1}{2}}\ \mathrm{Th}^{\frac{1}{2}}$
$\mathrm{Sa}^{\frac{4}{5}}\ \mathrm{Su}^{\frac{1}{2}}$
$\mathrm{F}^{\frac{1}{3}}$
$\mathrm{We}^{\frac{3}{4}}\ \mathrm{Sa}^{1}$

Figure 7.7: A predictable network. This graph gives the contact profiles of the nodes $a, b, c, d, e \in \mathcal{N}$. Unconnected nodes never meet each other: they have a null contact profile (and a corresponding delivery distribution $D_\perp$). The label connecting the other nodes describes which days they might have a contact. For example, there is one chance out of four that b and c meet on Thursday, and one out of two on Friday.

7.7.2 Guarantees

Our aim is now to find a way to deliver bundles that fulfills a given condition C as specified in definition 7.4, while trying to minimize the network's bandwidth/energy/memory consumption.

Ideally, the DTN is predictable enough to enforce condition C without duplicating any bundle. We thus propose to rely on algorithm 7.1 to find a first delivery scheme (and, thus, a first delivery distribution D_1).

If C is not fulfilled by D_1, we search for another fast bundle forwarding scheme using algorithm 7.1; let D_2 be its delivery distribution. We then duplicate the bundle on both delivery schemes, yielding a distribution $D_1 \oplus D_2$. We have already pointed out that $D_1 \oplus D_2 \succeq D_1$, thus $C(D_1 \oplus D_2)$ is more likely to be *true* then $C(D_1)$.

This process is iterated until C is finally fulfilled.

As mentioned in section 7.5.2, the distribution computed by the "duplication" ($\oplus$) operator is biased if its operands are not *independent* distributions. The simple distribution formula $(D_{sb} \otimes D_{bd}) \oplus (D_{sb} \otimes D_{bd})$ brings to light the problem caused by dependent distributions.

To avoid this bias, we ensure that D_1 and D_2 are independent by forbidding D_2 to rely on the nodes involved in D_1 (source and destination nodes excluded, line 5).

The resulting algorithm is given below.

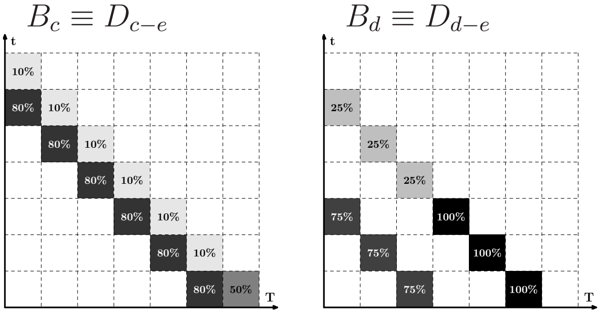
Nothing guarantees of course that there exists a way to deliver bundles that satisfies C: even an epidemic broadcasting might not suffice.

1 $B_a \equiv B_b \equiv D_\perp$

$B_c \equiv B_d \equiv D_\perp$

$B_e \equiv D_\top$

2



These plots show how our probabilistic Bellman-Ford algorithm behaves. This example is based on a simple network made of 5 nodes. The nodes contact profiles are given in figure 7.7. In this example, a is the source node and e is the destination.

At first (**1**), all the nodes (but the destination) have no knowledge of any path to the destination; their best distribution is thus set to $D_\perp$. The destination's delivery distribution to itself is of course $D_\top$.

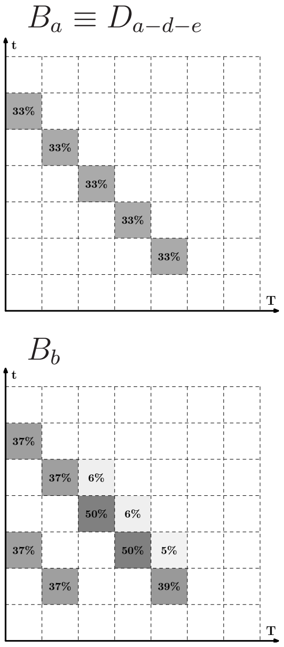
After the first iteration (**2**), since only c and d have contacts with the destination, only B_c and B_d are modified. They are set to the direct contact with the destination distribution since, for example, $D_{c-e} \otimes D_\top = D_{c-e}$. The delivery distributions are depicted as a square plot; the x-axis is the bundle sending time, the y-axis is the delay to reach the destination. Each square represents a 24 hour period, the first column matches bundles sent on Monday.

During the next iteration (**3**), a discovers it might meet with d before d meets e. B_a is thus changed to $D_{a-d} \otimes D_{d-e}$. The bundles received by b can be forwarded to c or d. The distributions D_{b-c-e} and D_{b-d-e} are thus compared; bundles sent Tuesday or before are sent *via* d, those sent after Tuesday are sent *via* c.

The last iteration (**4**) allows a to decide when bundles should be sent to b or d. The distributions B_a and $D_{a-b} \otimes B_b$ are thus compared. Neither c nor d should forward bundles to b, thus B_c and B_d are left untouched.

The algorithm is stabilized since neither b, c, or d should forward bundles *via* a.

3



4

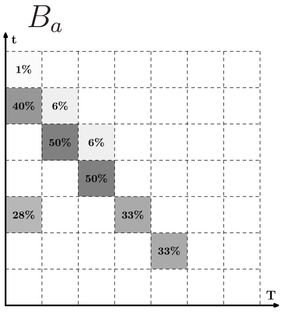


Figure 7.8: Finding a shortest path using the probabilistic Bellman-Ford algorithm.

Algorithm 7.2: Constrained probabilistic delivery

```
  Data: Network nodes $\mathcal{N}$; delivery condition $C$
  Data: Bundle source $s$ and destination $d$
1 $B \leftarrow D_\perp$ ;
2 repeat
3     Using nodes in $\mathcal{N}$, compute $D \in \mathcal{D}$ via algorithm 7.1 ;
4     $B \leftarrow B \oplus D$ ;
5     $\mathcal{N} \leftarrow \big(\mathcal{N} \setminus \{\text{nodes involved in } D\}\big) \cup \{s, d\}$ ;
6 until $C(B)$ or $\mathcal{N} = \{s, d\}$ ;
```

7.7.3 More on disjoint delivery schemes

The *constrained probabilistic delivery* algorithm above computes a delivery scheme that consists of duplicating the bundle to multiple, independent, non-duplicating delivery schemes.

To ensure independence, algorithm 7.2 enforces those non-duplicating delivery schemes to operate on completely distinct node sets. This might be too stringent if the network is small or sparse. We thus propose to allow such a delivery scheme to use nodes that are *unlikely* to receive a bundle according to the other schemes. The resulting delivery distributions will thus be *almost* independent.

Line 5 of algorithm 7.2 is thus changed: only the nodes involved in D with a probability higher than a given threshold are removed. The specific value of this threshold is a function of the network considered.

The rest of this section explains how to compute the probability that a given node receives a bundle, given a (non-duplicating) delivery scheme computed by algorithm 7.1.

We have seen that the proposed modified Bellman-Ford algorithm does not lead to a simple routing table: if a bundle reaches a given node at time T, its next hop depends on its destination *and on* T. Each node n divides time in intervals $I_1^n, I_2^n, \ldots$ (by means of the $\oslash$ operator, algorithm 7.1 line 7), and each interval I matches a given next hop $H_n(I)$. In the example figure 7.8, node b has defined two intervals: $I_1^b = [\text{Monday}, \text{Tuesday}]$ and $I_2^b = [\text{Wednesday}, \text{Sunday}]$; $H_b(I_1^b) = d$ and $H_b(I_2^b) = c$.

A bundle crosses a number of nodes on its way to its destination. We compute the probability P_n that a given node n is one of them.

Let s be the bundle source node and d the destination. If the bundle is ready to be sent at time T, it should reach $n = H_s(I)$, where I is the time interval of s such that $T \in I$. The bundle arrival time at n follows the contact distribution $N(x) = D_{sn}(T, x)$, thus $P_n = \int_0^\infty N(x)\,dx$.

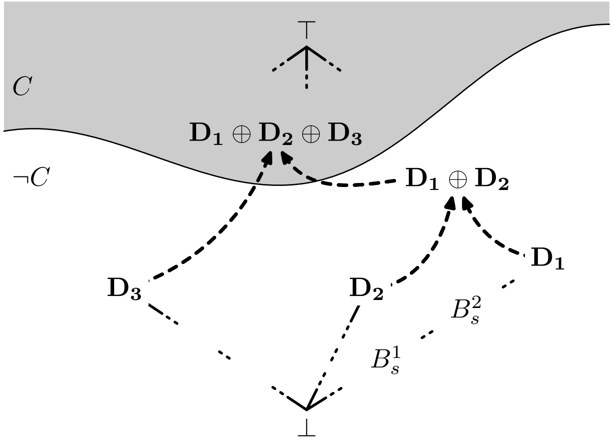

⊤
C
$\mathbf{D_1 \oplus D_2 \oplus D_3}$
¬C
$\mathbf{D_1 \oplus D_2}$
$\mathbf{D_1}$
$\mathbf{D_3}$
$\mathbf{D_2}$
B_s^2
B_s^1
⊥

Figure 7.9: Finding a fast delivery scheme that fulfills condition C using algorithms 7.1 and 7.2. This figure represents the delivery distribution lattice (introduced in figure 7.5); it is depicted the usual way (basically, an element is greater than another one if it is placed above). The greyed area corresponds to elements that satisfy condition C. **1.** The adapted Bellman-Ford algorithm is used to find a distribution (D_1) that characterizes a fast way to deliver bundles; B_s^i denotes the source node's best distribution found after i iterations. We have $\perp \preceq B_s^1 \preceq B_s^2 \preceq \cdots \preceq D_1$. **2.** Since $\neg C(D_1)$, another disjoint delivery distribution, D_2, is computed using algorithm 7.1. Combined with D_1, it leads to $D_1 \oplus D_2$ which still does not satisfy C. D_3 is thus computed, and combined with D_1 and D_2, gives a satisfactory delivery scheme $D_1 \oplus D_2 \oplus D_3$. We have $D_1 \preceq D_1 \oplus D_2 \preceq D_1 \oplus D_2 \oplus D_3$.

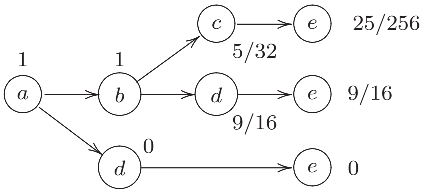
Once the bundle has been received by n, each time interval I_i^n matches a potential next hop $n_i = H_n(I_i^n)$. We have:

$$N_i^n(x) = \int_{\{t \in I_i^n | t \leq x\}} N(t)\, D_{nn_i}(t, x-t)\, dt \tag{7.10}$$

$$P_{n_i} = \int_0^\infty N_i^n(x)\, dx \tag{7.11}$$

Equation (7.10) gives the bundle time arrival distribution at the next hop n_i. This process can be continued recursively until the probability of receiving the bundle is known for all nodes.

The bundle forwarding process can be represented by a graph. The children of a node are the potential next hops. The graph obtained for the example depicted in figure 7.8 is given below.



The numbers labelling the nodes are the probabilities of receiving the bundle, as given by (7.11). The destination e thus receives the bundle with a probability of $\frac{25}{256} + \frac{9}{16}$.

7.8 Conclusion and future works

We propose to model contacts between a disruption tolerant network's mobile nodes as a random process, characterized by contact distributions. Such a description is more general than those generally encountered in the literature.

We show how such contact distributions can be combined to compute the bundle delivery delay distribution corresponding to a given delivery strategy (*i.e.* a description of the nodes forwarding decisions). We show how the Bellman-Ford algorithm can be adapted to cope with such stochastic networks.

There is a tradeoff between a bundle's delivery probability/delay and the consumption of network resources. We propose to duplicate bundles along disjoint "shortest" paths so as to meet a given delivery guarantee without consuming too many resources. The corresponding algorithms are given.

The work presented in this chapter can be continued along several lines.

We have proposed a way to route bundles through the network; other routing strategies should be explored and compared.

Three operators on delivery distributions have been defined. Others could be added so as to describe more subtle routing decisions, or to deal with bundles transmission delays.

Network traces should be analysed so as to quantify their predictability, to compare delivery strategies, and to measure how predictability impacts performance. In this context, both the Reality ([EP05, EP06]) and Haggle ([Dio]) projects provide valuable contact traces collected in real-world situations.

8 Conclusions & Perspectives

We can only see a short distance ahead,
but we can see plenty there that needs to be done.

— ALAN M. TURING,
Computing Machinery and Intelligence (1950)

Mobile networks have become nearly ubiquitous and now propose a decent bandwidth to always smaller and cheaper wireless devices.

Users have been quick to endorse the technology and now demand an on-the-go access to interactive services such as voice, video on demand, or teleconferencing over the Internet.

In this context, mobility prediction has an important role to play. It allows the network to take proactive actions that mitigate the impact of user mobility.

In the present thesis, we have seen the mobility prediction problem from three points of view that we can summarize using three questions:

- *What is the practicability of mobility prediction?* Prediction supposes a certain regularity of the terminal movements. This question explores the validity of this assumption.
- *How can we perform it?* If the terminal movements have a degree of regularity, how can we observe and model it so as to actually predict the mobile's destination?
- *What benefit does it bring?* If we suppose an effective mobility prediction scheme, how can it be used to improve the network performance?

In the following, we give a quick overview of the answers we have given to those questions and how the undertaken researches could be continued.

Mobility prediction can be performed either by the network infrastructure or by the terminals themselves. While it might seem that this last solution — even

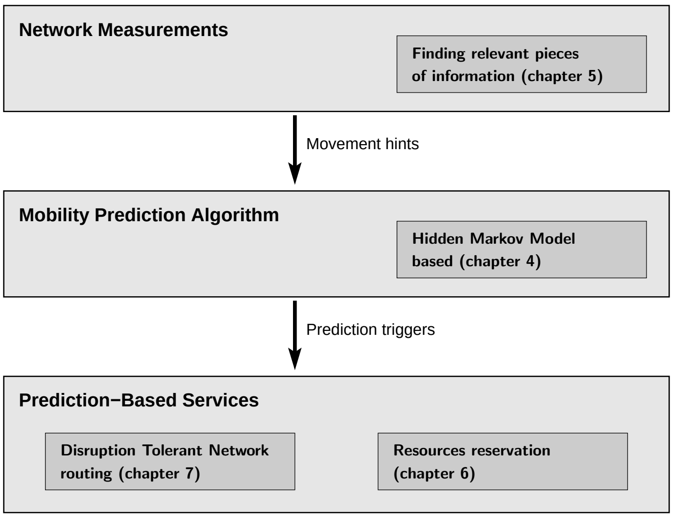

Network Measurements
Finding relevant pieces of information (chapter 5)
Movement hints
Mobility Prediction Algorithm
Hidden Markov Model based (chapter 4)
Prediction triggers
Prediction−Based Services
Disruption Tolerant Network routing (chapter 7)
Resources reservation (chapter 6)

Figure 8.1: Overview of the thesis.

if it appears to be more difficult to implement in practice — should lead to less prediction errors, we have in fact shown that, at least in the particular case of WLANs, the difference can be quite limited.

This comparison between prediction agents (access point or mobile terminal) could be continued with other network technologies. It would be interesting to perform similar experiments using information related to the mobile physical location. Since a lot of motion constraints (*e.g.* streets and roads) are shared by all the mobiles, our feeling is that the results we have obtained in the case of WLANs should continue to hold.

Different tasks (*e.g.* routing, resource reservation, pre-registration,. . .) can benefit from mobility prediction. In our view, it is thus interesting to see it as a general service. It is thus sensible to try and design generic prediction methods such as the one we proposed in chapter 4. It helps separate out technology-dependent concerns (such as locating the terminal) from the prediction algorithm itself, and removes a number of hypotheses about the process under study. For example, we do not assume the knowledge of the motion dynamic or the cell geometry.

Recently, several works emphasize the importance of “cognitive” networks. The idea is to automatically learn the workings of the network in order to help its management. This matches the way we view mobility prediction, and we argue that it fits well to this framework.

Cognitive networks output management hints, and take their inputs from the monitoring of the network itself. In chapter 5, we have presented a way to route the pieces of information extracted from the network monitoring in such a way that uninteresting measurements are discarded, while relevant ones are routed sensibly.

Even if we have presented this work in the (restrictive) context of mobility prediction, we think this is a first step towards completely decentralized, “intelligent" cognitive systems. One could for example imagine that unusual bandwidth consumption or packet loss measurements could be automatically sent to the appropriate decision point which would respond with new routing rules.

We have presented ways of using mobility prediction in chapters 6 and 7.

Chapter 6 deals with a call admission algorithm designed for cellular networks. Our contribution is twofold. Using simulations, we have first determined the impact of the prediction method's accuracy on the performance of the call admission algorithm. Secondly, we have pinpointed the problem induced by the presence of selfish users which could degrade the performance of the whole system for their own benefit; we have proposed a solution to restore fairness.

Such problems of fairness can also be addressed using the *theory of incentives* (which is derived from *game theory*). It would be interesting to compare this approach to the one we have proposed, and see their respective benefits and shortcomings.

Chapter 7 tackles the problem of packet routing in a Disruption Tolerant Network. Although the applicability of DTNs remains unclear, we think they might be a good network model for the swarm of small short range wireless devices that are now commonplace.

With DTNs, packet forwarding depends on changes in the network topology, which themselves depends on nodes mobility. Under the assumption that node mobility can be (at least roughly) predicted, and assuming the packet forwarding strategy is known, we have built a framework that allows us to estimate the packet end-to-end delay distribution. In the future, we hope it will help realize studies in a formal context and to ease the understanding of new routing algorithms.

Using the above-mentioned framework, we have been able to elegantly modify the well-known Bellman-Ford algorithm in order to suit it to disruption tolerant networks. The resulting procedure generalizes other works that have been done in the same context.

As a future work, new routing algorithms should be developed, and their performance should be evaluated using real mobility traces. Considering a network of devices that can communicate if their owner are next the other, the network topology is then directly determined by social behaviour. An interesting research area is thus the relationship between packet routing and social networks. The impact of mobility prediction accuracy should be quantified.

A

Acronyms

AAA — Authentication, Authorization and Accounting

ACK — Acknowledgment

AI — Artificial Intelligence

AP — Access Point

APC — Access Point Centric

AR — Access Router

ARP — Address Resolution Protocol

ATM — Asynchronous Transfer Mode

BF — Bellman-Ford

BS — Base Station

BW — Bandwidth

CAC — Call Admission Control

CAR — Candidate Access Router

CARD — Candidate Access Router Discovery

CDMA — Code Division Multiple Access

CN — Corresponding Node

CTT — Cell Travelling Time

CoA — Care-of Address

DE — Decision Engine

DS — Distribution System

DSL — Digital Subscriber Liner

DTN — Disruption/Delay Tolerant Network

FA — Foreign Agent

FBack — Fast Binding Acknowledgment

FBU — Fast Binding Update

FIFO — First In First Out

FHO — Fast Handover

FN — Foreign Network

GFA — Gateway Foreign Agent

GPS — Global Positioning System

GSM — Global System for Mobile communication

HA — Home Agent

HAck — Handover Acknowledge

HI — Handover Initiate

H-MIP — Hierarchical Mobile IP

HMM — Hidden Markov Model

HO — Handover

ICMP — Internet Control Message Protocol

IETF — Internet Engineering Task Force

IEEE — Institute of Electrical and Electronics Engineers

IrDA — Infrared Data Association

LAN — Local Area Network

LZ — Lempel-Ziv

MA — Mobility Agent

MAC — Medium Access Control

MAN — Metropolitan Area Network

MANET — Mobile Ad Hoc Network

MAP — Mobile Anchor Point

MH — Mobile Host

MHC — Mobile Host Centric

MIP — Mobile IP

ML — Machine Learning

MN — Mobile Node

MS — Mobile Station

MT — Mobile Terminal

NAR — Next Access Router

NCoA — New CoA

NPC — Non-deterministic Polynomial-time Complete

PAR — Previous Access Router

PCoA — Previous CoA

PCR — Predictive Channel Reservation

PCS — Personal Communication System

PDA — Personal Digital Assistant

PPM — Prediction by partial match

QoS — Quality of Service

RARP — Reverse ARP

RFC — Request for Comments

RSSHT — Relative Signal Strength with Hysteresis and Threshold

RTT — Round-Trip Time

SCTP — Stream Control Transport Protocol

SIP — Session Initiation Protocol

S-MIP — Seamless Mobile IP

SNMP — Simple Network Management Protocol

SNR — Signal to Noise Ratio

SS — Signal Strength

STA — Mobile station

TDOA — Time Difference of Arrival

TCP — Transmission Control Protocol

UDP — User Datagram Protocol

UMTS — Universal Mobile Telecommunications System

VoIP — Voice over IP

WCDMA — wideband CDMA

WLAN — Wireless LAN

B

Implementation of HMMs-related algorithms

In chapter 4, we have extensively presented and used *Hidden Markov Models*, which is a simple learning method that allows to model various processes under the first-order *markovian* hypothesis — see equation (4.1) — and assuming the process is time-independent.

Those models have been introduced by Rabiner and Juang, and are frequently used, for example, in biology (*e.g.* [FPD$^+$06]) and speech recognition (*e.g.* [RJ93]).

We have implemented an efficient, open-source[1], clean and general purpose library written in the Java$^{\text{TM}}$ language ([JGB05]) and called *Jahmm*[2].

It has been cited in various research articles and is used as an educational tool for several machine learning courses ([ZBRC06, LC05, LH05, CKP$^+$06]...).

It covers all the standard algorithms related to HMMs: forward-backward (estimation of the probability of a sequence), Viterbi (computation of the most likely state sequence given an observation sequence), Kullback-Leibler (probabilistic distance measure between HMMs, see [RJ85]), generation of observation sequences given a HMM, Baum-Welch and k-means (the last two are learning procedures). The algorithms complexities are guaranteed to be those given by the theory.

Two difficulties have been encountered during the library programming that are seldom discussed in theory:

- The Baum-Welch training algorithm involves evaluating the probability of the learning observation sequence. This can cause a problem for long sequences since the probability of a sequence decreases exponentially fast with its length. This problem can be solved by an appropriate scaling some variables involved in this computation (see [RJ93]).
- The training algorithms must handle multiple observation sequences. The

[1] It is distributed under the GNU General Public License v2, see `http://www.gnu.org/`.

[2] For *Java HMM*, pronounced *jam* (see [Fra03]).

training is usually done using a single observation sequence. In practice, several observation sequences $S_1, S_2, \ldots, S_n$ are used to estimate the model parameters. The Baum-Welch and k-means procedures must thus be modified so as to maximize $\prod_{i=1}^{n} \mathrm{P}[S_i \mid \lambda]$ or $\prod_{i=1}^{n} \mathrm{P}[S_i, Q{*}_i \mid \lambda]$ respectively — with the same notations as those introduced in equation (4.9).

This library is available at
`http://www.run.montefiore.ulg.ac.be/~francois/software/jahmm/`

It is one of the most well-known general purpose HMM library available in open-source.

C

Properties of Predictable DTNs

We have seen in chapter 7 that the contacts between the networks's nodes can be formalized by means of first contact distributions (the set of such distributions is denoted $\mathcal{C}$, see definition 7.1) and delivery distributions (the delivery distributions set is denoted $\mathcal{D}$, see definition 7.2).

Both the elements of $\mathcal{C}$ and $\mathcal{D}$ can be compared by means of the $\preceq$ (partial) order relation (definition 7.3).

This appendix details a few interesting properties of this operator and of the elements of $\mathcal{C}$ and $\mathcal{D}$.

The definition of the operators that can be applied to delivery distributions are given in section 7.5.

Lemma C.1. *$\forall D_1, D_2, D_3 \in \mathcal{D}$, we have $D_2 \succeq D_3 \Rightarrow D_1 \oplus D_2 \succeq D_1 \oplus D_3$.*

Proof. Given the definition of $\oplus$ and $\succeq$, one must prove that, $\forall D_1, D_2 \in \mathcal{D}$, $\forall T \geq 0, t \geq 0$, given $D_2 \succeq D_3$:

$$\begin{aligned}&\int_0^t \left[1 - \int_0^x D_1(T,y)\,dy\right] D_2(T,x) + \left[1 - \int_0^x D_2(T,y)\,dy\right] D_1(T,x)\,dx \\ &\geq \int_0^t \left[1 - \int_0^x D_1(T,y)\,dy\right] D_3(T,x) + \left[1 - \int_0^x D_3(T,y)\,dy\right] D_1(T,x)\,dx\end{aligned} \tag{C.1}$$

The left-hand part can be written as:

$$\begin{aligned}&\int_0^t D_1(T,x)\,dx + \int_0^t D_2(T,x)\,dx - \\ &\qquad \int_0^t \int_0^x \left[D_1(T,x)D_2(T,y) + D_1(T,y)D_2(T,x)\right] dy\,dx\end{aligned} \tag{C.2}$$

Changing the double integral's integration order, the last term of (C.2) is equal to:

$$\int_0^t D_1(T,x) \int_0^x D_2(T,y)\, dy\, dx + \int_0^t D_1(T,x) \int_x^t D_2(T,y)\, dy\, dx = \int_0^t D_1(T,x)\ dx \int_0^t D_2(T,y)\, dy \quad \text{(C.3)}$$

The same procedure can be applied to the right-hand part of (C.1). (C.1) is thus equivalent to:

$$\int_0^t D_2(T,x)\, dx \geq \int_0^t D_3(T,x)\, dx \quad \text{(C.4)}$$

Which holds by hypothesis. □

Corollary C.1. *$\forall D_1, D_2 \in \mathcal{D}$, we have $D_1 \oplus D_2 \succeq D_1$.*

Proof. From Lemma C.1, $D_1 \oplus D_2 \succeq D_1 \oplus D_\perp = D_1$. □

Bibliography

[AZ01] A. Aljadhai and T.F. Znati. Predictive mobility support for QoS provisioning in mobile wirelessenvironments. *IEEE Journal on Selected Areas in Communications*, 19(10):1915–1930, October 2001. (Cited on page 17.)

[BB05] Mauro Brunato and Roberto Battiti. Statistical learning theory for location fingerprinting in wireless LANs. *Computer Networks*, 47(6):825–845, 2005. (Cited on page 21.)

[BBC$^+$98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. RFC 2475 (Informational), December 1998. Updated by RFC 3260. (Cited on page 2.)

[BBD03] N. Banerjee, K. Basu, and S. Das. Handoff delay analysis and measurement in SIP-based mobility management in wireless networks. In *International Parallel and Distributed Processing Symposium*, pages 224–231, April 2003. (Cited on page 7.)

[BBL05] Brendan Burns, Oliver Brock, and Brian Neil Levine. *MV* routing and capacity building in Disruption Tolerant Networks. In *Proc. IEEE INFOCOM*, pages 398–408, March 2005. (Cited on pages 111 and 112.)

[BCS94] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. RFC 1633 (Informational), June 1994. (Cited on page 2.)

[BD99] A. Bhattacharya and S. Das. LeZi-update: an information-theoretic approach to track mobile users in PCS networks. In *Proc. of ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, Seattle, August 1999. (Cited on pages 17, 19, 20, 35, 70, 87, and 92.)

[BD02] Amiya Bhattacharya and Sajal K. Das. LeZi-update: an information-theoretic framework for personal mobility tracking in PCS networks. *Wireless Networks*, 8(2/3):121–135, 2002. (Cited on pages 17, 20, and 28.)

[BEJ97] Jens Biesterfeld, Elyes Ennigrou, and Klaus Jobmann. Neural networks for location prediction in mobile networks. In *Proc. of the International Workshop on Applications of Neural Networks to Telecommunications*, page 207, 1997. (Cited on page 21.)

[Bel95] John S. Belrose. Fessenden and Marconi: Their differing technologies and transatlantic experiments during the first decade of this century, September 1995. International Conference on 100 Years of Radio. (Cited on page 1.)

[Bet01] C. Bettstetter. Smooth is better than sharp: a random mobility model for simulation of wireless n etworks. In *Proceedings of ACM Workshop on Modeling, Analysis and Simulation of Wireles s and Mobile Systems*, pages 19–27, Rome, Italy, July 2001. (Cited on page 68.)

[BH04] N. Baghaei and R. Hunt. Review of quality of service performance in wireless LANs and 3G multimedia application services. *Elsevier, Computer Communications*, 27(17):1684–1692, November 2004. (Cited on page 93.)

[BHT$^+$03] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking - an approach to interplanetary internet. *IEEE Communications Magazine*, 41(6):128–136, 2003. (Cited on page 110.)

[BR02] C.I. Bauer and S.J. Rees. Classification of handover schemes within a cellular environment. In *Proceedings of the 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 5, pages 2199–2203, September 2002. (Cited on page 3.)

[Cal00] P. Calhoun. Agent assisted hand-off. Draft, November 2000. `draft-ietf-mobileip-proactive-fa-03.txt`. (Cited on page 24.)

[Cas98] Claude Castelluccia. A hierarchical mobile IPv6 proposal. Technical report, INRIA TR-226, November 1998. http://www.inrialpes.fr/Planete/people/ccastel/. (Cited on page 6.)

[CB00] Ming-Hsing Chiu and Mostafa Bassiouni. Predictive schemes for handoff prioritization in cellular networks based on mobile positioning. *Journal on selected areas in communications*, 18(3):510–522, March 2000. (Cited on pages 28, 92, 93, 94, and 96.)

[CB04] J. Capka and R. Boutaba. Mobility prediction in wireless networks using neural networks. In *Proc. of the IFIP/IEEE international Conference on the Management of Multimedia Networks and Services (MMNS'04)*, October 2004. (Cited on page 21.)

[CBD02] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for Ad Hoc network research. *WCMC: Special Issue on Mobile ad hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002. (Cited on page 68.)

[CG$^+$00] A. T. Cambell, J. Gomez, et al. Cellular IP. Internet draft, IETF, January 2000. draft-ietf-mobileip-cellularip-00.txt. (Cited on page 6.)

[CGKW02] A. Campbell, J. Gomez, S. Kim, and C. Wan. Comparison of IP micro-mobility protocols. *IEEE Wireless Communications*, pages 72–82, February 2002. (Cited on page 6.)

[Che96] Stuart Cheshire. Latency and the quest for interactivity. White Paper, 1996. `http://www.stuartcheshire.org/papers/LatencyQuest.html`. (Cited on page 2.)

[Che03] Yue Chen. *Soft Handover Issues in Radio Resource Management for 3G WCDMA Networks*. PhD thesis, Queen Mary, University of London, September 2003. (Cited on page 93.)

[CHP96] D. Cong, M. Hamlen, and C. Perkins. The Definitions of Managed Objects for IP Mobility Support using SMIv2. RFC 2006 (Proposed Standard), October 1996. (Cited on page 5.)

[CKP$^+$06] H. J. Cha, Y. S. Kim, S. H. Park, T. B. Yoon, Y. M. Jung, and J. H. Lee. Learning styles diagnosis based on user interface behaviors for the customization of learning interfaces in an intelligent tutoring system. In *Lecture Notes in Computer Science*, volume 4053/2006, pages 513–524. Springer Berlin, 2006. (Cited on pages 53 and 135.)

[CLF06] Vania Conan, Jeremie Leguay, and Timur Friedman. The heterogeneity of inter-contact time distributions: its importance for routing in delay tolerant networks, 2006. (Cited on page 38.)

[CLSS00] J. Chan, Björn Landfeldt, A. Seneviratne, and P. Sookavatana. Integrating mobility prediction and resource pre-allocation into a home-proxy based wireless Internet framework. In *Proc. of the 8th IEEE International Conference on Networks (ICON'00)*, page 18, Washington, DC, USA, 2000. IEEE Computer Society. (Cited on page 17.)

[CPRW03] David D. Clark, Craig Partridge, J. Christopher Ramming, and John T. Wroclawski. A knowledge plane for the Internet. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–10, New York, NY, USA, 2003. ACM Press. (Cited on pages 21 and 87.)

[CS98a] J.J. Caffery and G.L. Stuber. Overview of radiolocation in CDMA cellular systems. *Communications Magazine*, 36(4):38–45, April 1998. (Cited on page 23.)

[CS98b] Sunghyun Choi and Kang G. Shin. Predictive and adaptive bandwidth reservation for hand-offs in QoS-sensitive cellular networks. *ACM SIGCOMM Computer Communication Review*, 28(4):155–166, 1998. (Cited on pages 44, 49, 61, 64, and 92.)

[CS99] Jonathan Chan and Aruna Seneviratne. A practical user mobility prediction algorithm for supporting adaptive QoS in wireless networks. *icon*, 00:104, 1999. (Cited on pages 17 and 33.)

[CTI06] CTIA. Mobile phones grow even more popular. Semi-annual wireless industry survey, 2006. `http://www.ctia.org/research_statistics/index.cfm/AID/10030`. (Cited on page 1.)

[CW84] John G. Cleary and Ian H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, 1984. (Cited on page 35.)

[CZD$^+$04] Roberto Corvaja, Andrea Zanella, Michele Dossi, Adolfo Tontoli, and Paolo Zennaro. Experimental performance of the handover procedure in a WiFi network. In *Proceedings of WPMC'04*, September 2004. (Cited on page 3.)

[CZS98] J. Chan, S. Zhou, and A. Seneviratne. A QoS adaptive mobility prediction scheme for wireless networks. *IEEE Global Telecommunications Conference, 1998. GLOBECOM 98.*, 3:1414–1419, 1998. (Cited on pages 17 and 33.)

[Dee91] S. Deering. ICMP Router Discovery Messages. RFC 1256 (Proposed Standard), September 1991. (Cited on page 6.)

[Dio] Christophe Diot. The Haggle project. `http://www.haggleproject.org/`. (Cited on page 126.)

[DL03] Tom Dietterich and Pat Langley. Machine learning for cognitive networks: Technology assessment and research challenges. `http://www.cs.umd.edu/~clancy/docs/wcm-crnet06.pdf`, May 2003. (Cited on page 21.)

[EP05] N. Eagle and A. Pentland. Eigenbehaviors: Identifying structure in routine. Technical Report 601, MIT Media Lab Vision and Modeling, 2005. (Cited on page 126.)

[EP06] N. Eagle and A. Pentland. Reality mining: Sensing complex social systems. In *Personal and Ubiquitous Computing*, volume 10-4, 2006. (Cited on page 126.)

[ESE+01] F. Erbas, J. Steuer, D. Eggesieker, K. Kyamakya, and K. Jobinann. A regular path recognition method and prediction of user movements in wireless networks. In *Vehicular Technology Conference, 2001. VTC 2001 Fall. IEEE VTS 54th*, volume 4, pages 2672–2676, 2001. (Cited on page 21.)

[FHL06] Xiaoming Fu, Dieter Hogrefe, and Deguang Le. A review of mobility support paradigms for the Internet. *IEEE Communications Surveys & Tutorials*, 8(1):38–51, January 2006. (Cited on page 5.)

[FL05a] Jean-Marc François and Guy Leduc. Entropy-based knowledge spreading and application to mobility prediction. In *Proc. of ACM International conference on Emerging Network Experiments and Technologies (Co-Next)*, Toulouse, France, October 2005. ACM Press. (Cited on page 10.)

[FL05b] Jean-Marc François and Guy Leduc. Mobility prediction's influence on QoS in wireless networks: a study on a call admission algorithm. In *Proc. of 3rd International Symposium on Modeling and Optimization in Mobile, ad hoc and Wireless Networks (WiOpt)*, Trentino, Italy, April 2005. IEEE Press. (Cited on page 10.)

[FL06a] Jean-Marc François and Guy Leduc. Predictable disruption tolerant networks and delivery guarantees. Technical Report arXiv:cs.NI(0612034v1), University of Liège, December 2006. (Cited on page 10.)

[FL06b] Jean-Marc François and Guy Leduc. Prédiction de mobilité par le mobile ou par le point d'accès: comparaison sur base de traces réelles. In *Actes de Colloque Francophone sur l'Ingénierie des Protocoles (CFIP)*, Tozeur, Tunisie, Octobre 2006. Lavoisier. (Cited on pages .)

[FL07a] Jean-Marc François and Guy Leduc. AP and MN-centric mobility prediction: a comparative study based on wireless traces. In IFIP, editor, *LNCS Proc. of Networking 2007*. Springer-Verlag, May 2007. (Cited on page 10.)

[FL07b] Jean-Marc François and Guy Leduc. Delivery guarantees in predictable delay tolerant networks. In IFIP, editor, *LNCS Proc. of Networking 2007*. Springer-Verlag, May 2007. (Cited on page 10.)

[FLM03] Jean-Marc François, Guy Leduc, and Sylvain Martin. évaluation d'une méthode de prédiction des déplacements de terminaux dans les réseaux mobiles. In *Actes de Colloque Francophone sur l'Ingénierie des protocoles (CFIP)*, pages 189–202, Paris, France, Octobre 2003. (Cited on page 10.)

[FLM04] Jean-Marc François, Guy Leduc, and Sylvain Martin. Learning movement patterns in mobile networks: a generic approach. In *Proc. of European Wireless 2004*, pages 128–134, Barcelona, Spain, February 2004. (Cited on pages 10 and 92.)

[FP03] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003. (Cited on page 23.)

[FPC05] H. Fathi, R. Prasad, and S. Chakraborty. Mobility management for VoIP in 3G systems: evaluation of low-latency handoff schemes. *IEEE Wireless Communications*, 12(2):96–104, April 2005. (Cited on page 6.)

[FPD$^+$06] T. Friedrich, B. Pils, T. Dandekar, J. Schultz, and T. Muller. Modelling interaction sites in protein domains with interaction profile hidden markov models. In *Bioinformatics*, volume 22, pages 2851–2857, December 2006. (Cited on page 135.)

[Fra] Beth Frasco. Enhanced observed time difference (e-otd). `http://www.fcc.gov/911/enhanced/releases/aerial.pdf`. (Cited on page 23.)

[Fra03] Jean-Marc François. Jahmm: A Hidden Markov Model library, 2003. `http://www.run.montefiore.ulg.ac.be/~francois/software/jahmm`. (Cited on page 135.)

[Fra04] Jean-Marc François. Étude de mécanismes de prédiction de mouvement dans les réseaux mobiles. Master's thesis, Département d'Électricité, Électronique et Informatique, Université de Liège, 2003–2004. En vue de l'obtention du Diplôme d'Études Approfondies en Informatique. (Cited on page 10.)

[Fre95] Yoav Freund. Boosting a weak learning algorithm by majority. *Inf. Comput.*, 121(2):256–285, 1995. (Cited on page 73.)

[Geu02] P. Geurts. *Contributions to decision tree induction: bias/variance tradeoff and time series classification*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Liège, May 2002. (Cited on page 73.)

[GJP02] Eva Gustafsson, Annika Jonsson, and Charles E. Perkins. Mobile ipv4 regional registration. Draft, October 2002. `draft-ietf-mobileip-reg-tunnel-07.txt`. (Cited on page 6.)

[GPZ05] Francisco González, Jesús A. Pérez, and Victor H. Zárate. HAMS: Layer 2 handoff accurate measurement strategy in WLANs 802.11. In *Proceedings of the First Workshop on Wireless Network Measurements (WiNMee'05)*, Trentino, Italy, April 2005. (Cited on page 4.)

[Hay99] Victor Hayes. *LAN — Specific requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. ANSI/IEEE, 1999. (Cited on page 3.)

[HGPC99] Xiaoyan Hong, Mario Gerla, Guangyu Pei, and Ching-Chuan Chiang. A group mobility model for Ad Hoc wireless networks. In *Proceedings of MSWiM'99*, Seattle, WA, August 1999. (Cited on page 68.)

[HKA04] Tristan Henderson, David Kotz, and Ilya Abyzov. The changing usage of a mature campus-wide wireless network. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 187–201, New York, NY, USA, 2004. ACM Press. (Cited on page 34.)

[HM99] Stathes Hadjiefthymiades and Lazaros Merakos. ESW4: enhanced scheme for WWW computing in wireless communication environments. *ACM SIGCOMM Computer Communication Review*, 29(5):24–35, 1999. (Cited on pages 21, 49, and 61.)

[HM02] Stathes Hadjiefthymiades and Lazaros Merakos. Using path prediction to improve TCP performance in wireless/mobile communications. *IEEE Communications Magazine*, 40(8):54–61, August 2002. (Cited on pages 21 and 33.)

[HSSEM02] R. Hsieh, A. Seneviratne, H. Soliman, and K. El-Malki. Performance analysis on Hierarchical Mobile IPv6 with fast-handoff over end-to-end TCP. In *Proc. of IEEE Global Telecommunications Conference (GLOBECOM'02)*, volume 3, pages 2488–2492, November 2002. (Cited on page 26.)

[HZS03] R. Hsieh, Z. Zhou, and A. Seneviratne. S-MIP: A seamless handoff architecture for Mobile IP. In *Proceedings of IEEE Infocom'03*, volume 3, pages 1774–1784, April 2003. (Cited on pages 15 and 16.)

[IET] IETF. Ietf request for comments. http://www.ietf.org/rfc.html. (Cited on page 24.)

[Inf06] Infonetics Research. Report on voice over IP, 2006. (Cited on page 2.)

[JGB05] Guy Steele James Gosling, Bill Joy and Gilad Bracha. *The Java^{TM}Language Specification Third Edition.* Addison-Wesley, 2005. http://java.sun.com/docs/books/jls/index.html. (Cited on page 135.)

[JLW05] Evan P. C. Jones, Lily Li, and Paul A. S. Ward. Practical routing in delay-tolerant networks. In *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 237–243, New York, NY, USA, 2005. ACM Press. (Cited on page 110.)

[JOW$^+$02] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *ASPLOS, San Jose, CA*, October 2002. (Cited on page 110.)

[JP03] D. Johnson and C. Perkins. Mobility support in IPv6. Internet draft, IEFT, February 2003. draft-ietf-mobileip-ipv6-21.txt. (Cited on page 5.)

[JR90] B. H. Juang and L. H. Rabiner. The segmental k-means algorithm for estimating the parameters of hidden markov models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(9):1639–1641, September 1990. (Cited on page 54.)

[JSA00] Philippe Jacquet, Wojciech Szpankowski, and Izydor Apostol. *An universal predictor based on pattern matching, preliminary results*, chapter 7, pages 75–85. Daniele Gardy and Abdelkader Mokkadem editors, Birkhauser, 2000. (Cited on page 35.)

[KE02] David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. In *MobiCom*, pages 107–118, September 2002. Revised and corrected as Dartmouth CS Technical Report TR2002-432. (Cited on pages 36 and 41.)

[KGL05] I. El Khayat, P. Geurts, and G. Leduc. Improving TCP in wireless networks with an adaptive machine-learnt classifier of packet loss causes. In *Proc. of IFIP International Networking Conference (Networking 2005)*, pages 549–560. LNCS, Springer-Verlag, May 2005. (Cited on page 21.)

[KKK01] Geng-Sheng Kuo, Po-Chang Ko, and Min-Lian Kuo. A probabilistic resource estimation and semi-reservation scheme for flow oriented multimedia wireless networks. *IEEE Communications Magazine*, 39(2):135–141, February 2001. (Cited on page 28.)

[KMR04] James Kempf, Peter McCann, and Phil Roberts. IP mobility and the CDMA Radio Access Network: Applicability statement for soft handoff. Draft, March 2004. `draft-kempf-cdma-appl-03.txt`. (Cited on pages 3 and 93.)

[Koo05] R. Koodli. Fast Handovers for Mobile IPv6. RFC 4068 (Experimental), July 2005. (Cited on page 24.)

[Kot05] David Kotz. CRAWDAD, a Community Resource for Archiving Wireless Data At Dartmouth. `http://crawdad.cs.dartmouth.edu/`, 2005. (Cited on page 36.)

[KP06] Rajeev Koodli and Charles E. Perkins. Mobile IPv4 fast handovers. Draft, October 2006. `draft-ietf-mip4-fmipv4-02.txt`. (Cited on page 24.)

[Laa05] Kari Laasonen. Clustering and prediction of mobile user routes from cellular data. *PKDD 2005, LNAI*, 3721:569–576, 2005. (Cited on pages 17, 20, and 33.)

[LAN95] David A. Levine, Ian F. Akyildiz, and Mahmoud Naghshineh. The shadow cluster concept for resource allocation and call admission in ATM-based wireless networks. In *Proc. of the 1st annual international conference on Mobile computing and networking (MobiCom'95)*, pages 142–150, New York, NY, USA, 1995. ACM Press. (Cited on pages 28 and 92.)

[LBC98] Tong Liu, P. Bahl, and I. Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks. *IEEE JSAC*, 16(6):922–936, August 1998. (Cited on pages 20, 23, 50, 70, and 92.)

[LC05] David Lindsay and Siân Cox. Effective probability forecasting for time series data using standard machine learning techniques. In *Lecture Notes in Computer Science*, volume 3686/2005, pages 35–44. Springer Berlin, 2005. (Cited on pages 53 and 135.)

[LDS03] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, 2003. (Cited on page 110.)

[LFC05] Jérémie Leguay, Timur Friedman, and Vania Conan. DTN routing in a mobility pattern space. In *WDTN '05: Proceeding of the 2005 ACM*

SIGCOMM workshop on Delay-tolerant networking, pages 276–283, New York, NY, USA, 2005. ACM Press. (Cited on page 110.)

[LH03] Ben Liang and Zygmunt J. Haas. Predictive distance-based mobility management for multidimensional pcs networks. *IEEE/ACM Trans. Netw.*, 11(5):718–732, 2003. (Cited on page 23.)

[LH05] Ulf Leser and Jorg Hakenberg. What makes a gene name? named entity recognition in the biomedical literature. *Brief Bioinform*, 6(4):357–369, 2005. (Cited on pages 53 and 135.)

[LM95] G.Y. Liu and G.Q. Maguire. A predictive mobility management algorithm for wireless mobile computing and communications. In *Proc. of international conference on Universal Personal Communications (ICUPC'95)*, Tokyo, Japan, November 1995. (Cited on pages 17, 20, and 92.)

[LSC$^+$05] M. Liebsch, A. Singh, H. Chaskar, D. Funato, and E. Shim. Candidate Access Router Discovery (CARD). RFC 4066 (Experimental), July 2005. (Cited on page 13.)

[LSG01] Sung-Ju Lee, William Su, and Mario Gerla. Wireless Ad Hoc multicast routing with mobility prediction. *Mobile Networks and Applications*, 6(4):351–360, August 2001. (Cited on page 92.)

[MAZ04] Shashidhar Merugu, Mostafa Ammar, and Ellen Zegura. Routing in space and time in networks with predictable mobility. Technical Report GIT-CC-04-7, Georgia Institute of Technology, 2004. (Cited on pages 110, 112, and 121.)

[MHM05] Mirco Musolesi, Stephen Hailes, and Cecilia Mascolo. Adaptive routing for intermittently connected mobile ad hoc networks. In *WOWMOM '05: Proceedings of the Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM'05)*, pages 183–189, Washington, DC, USA, 2005. IEEE Computer Society. (Cited on page 110.)

[Mit97] Thomas M Mitchell. *Machine Learning*. McGraw-Hill Higher Education, 1997. (Cited on page 21.)

[MK04] J. Manner and M. Kojo. Mobility Related Terminology. RFC 3753 (Informational), June 2004. (Cited on pages 3 and 8.)

[MN02] N. Montavont and T. Noel. Handover management for mobile nodes in IPv6 networks. *IEEE Communications Magazine*, 40(8):38–43, August 2002. (Cited on page 13.)

[MRD04] A. Misra, A. Roy, and S.K. Das. An information-theoretic framework for optimal location tracking in multi-system 4G wireless networks. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*, pages 286–297, 2004. (Cited on pages 17, 20, 33, and 70.)

[MS93] Andrew Myles and David Skellern. Comparing four IP based mobile host protocols. In *Selected papers of the 4th joint conference on European networking conference*, pages 349–355, Amsterdam, The Netherlands, 1993. North-Holland Publishing Co. (Cited on page 5.)

[MS03] Karim El Malki and Hesham Soliman. Simultaneous bindings for Mobile IPv6 fast handovers. Draft, May 2003. `draft-elmalki-mobileip-bicasting-v6-03.txt`. (Cited on page 8.)

[MSA03] Arunesh Mishra, Minho Shin, and William Arbaugh. An empirical analysis of the IEEE 802.11 MAC layer handoff process. *SIGCOMM Computing Communication Review*, 33(2):93–102, 2003. (Cited on pages 3 and 4.)

[NDD03] N. Nakajima, A. Dutta, and H. Das, S.and Schulzrinne. Handoff delay analysis and measurement for SIP based mobility in IPv6. In *IEEE International Conference on Communications*, volume 2, pages 1085–1089, May 2003. (Cited on page 7.)

[NPB03] Delphine Nain, Nosh Petigara, and Hari Balakrishnan. Integrated Routing and Storage for Messaging Applications in Mobile Ad Hoc Networks. In *WiOpt '03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Sophia-Antipolis, France, March 2003. (Cited on page 110.)

[NSA$^+$04] N. Niebert, A. Schieder, H. Abramowicz, G. Malmgren, J. Sachs, U. Horn, C. Prehofer, and H. Karl. Ambient networks — An architecture for communication networks beyond 3G. *Proceedings of IEEE Wireless Communications*, 11(2):14–22, April 2004. (Cited on page 2.)

[Per96a] C. Perkins. IP Encapsulation within IP. RFC 2003 (Proposed Standard), October 1996. (Cited on page 5.)

[Per96b] C. Perkins. IP Mobility Support. RFC 2002 (Proposed Standard), October 1996. Obsoleted by RFC 3220, updated by RFC 2290. (Cited on page 5.)

[Per96c] C. Perkins. Minimal Encapsulation within IP. RFC 2004 (Proposed Standard), October 1996. (Cited on page 5.)

[Per97] Charles E. Perkins. *Mobile IP*. Addison-Wesley, 1997. (Cited on page 5.)

[PGM04] Vijoy Pandey, Dipak Ghosal, and Biswanath Mokherjee. Exploiting user profile to support differentiated services in next-generation wireless networks. *IEEE Network*, pages 40–48, September/October 2004. (Cited on pages 70 and 92.)

[Qui93] Ross Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993. (Cited on pages 72, 74, and 86.)

[Rab89] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–282, February 1989. (Cited on pages 52, 53, and 54.)

[RB01] Pierre Reinbold and Olivier Bonaventure. A comparison of IP mobility protocols. Technical Report Infonet-2001-07, University of Namur, June 2001. (Cited on page 6.)

[RB03] Pierre Reinbold and Olivier Bonaventure. IP micro-mobility protocols. *IEEE Communications Surveys & Tutorials*, 5(1):40–57, 2003. (Cited on page 6.)

[RDM04] A. Roy, S.K. Das, and A. Mistra. Exploiting information theory for adaptive mobility and resource management in future cellular networks. *IEEE Wireless COmmunications*, 3(5):59–65, August 2004. (Cited on page 92.)

[RJ85] L. R. Rabiner and B. H. Juang. A probabilistic distance measure between hmms. *AT&T Technical Journal*, 64(2):391–408, February 1985. (Cited on page 135.)

[RJ86] L. R. Rabiner and B. H. Juang. An introduction to Hidden Markov Models. *IEEE ASSP Magazine*, 34-3:4–16, June 1986. (Cited on page 52.)

[RJ93] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. (Cited on page 135.)

[RLT$^+$00] R. Ramjee, T. La Porta, S. Thuel, et al. IP micro-mobility support using HAWAII. Internet draft, IETF, July 2000. draft-ietf-mobileip-hawaii-01.txt. (Cited on page 6.)

[RSC$^+$02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. Sip: Session initiation protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320. (Cited on page 7.)

[SBRJ02] C. Shen, G. Borkar, S. Rajagopalan, and C. Jaikaeo. Interrogation-based relay routing for ad hoc satellite networks. In *IEEE Globecom*, Taipei, Taiwan, November 17-21 2002. (Cited on page 110.)

[SH03] Tara Small and Zygmunt J. Haas. The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 233–244, New York, NY, USA, 2003. ACM Press. (Cited on page 110.)

[SK01] Wee-Seng Soh and Hyong S. Kim. Dynamic guard bandwidth scheme for wireless broadband networks. In *Proc. of IEEE Infocom'01*, pages 572–58, Anchorage, AK, April 2001. (Cited on pages 29, 33, and 92.)

[SK03] Wee-Seng Soh and Hyong S. Kim. QoS provisioning in cellular networks based on mobility prediction techniques. *IEEE Communications Magazine*, 41(1):86–92, January 2003. (Cited on pages 23 and 92.)

[SK04] Wee-Seng Soh and Hyong S. Kim. Dynamic bandwidth reservation in cellular networks using road topology based mobility predictions. In *Proc. of IEEE Infocom'04*, Hong Kong, March 2004. (Cited on pages 23, 33, 70, and 92.)

[SK05] Nancy Samaa and Ahmed Karmouch. A mobility prediction architecture based on contextual knowledge and spatial conceptual maps. *IEEE Trans. on mobile computing*, 4:537–551, November/December 2005. (Cited on pages 21 and 33.)

[SKJH04a] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. Evaluating location predictors with extensive Wi-Fi mobility data. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 1414–1424, March 2004. (Cited on pages 17, 35, 36, and 39.)

[SKJH04b] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. Evaluating location predictors with extensive Wi-Fi mobility data. Technical Report TR2004-491, Dept. of Computer Science, Dartmouth College, February 2004. (Cited on pages 34, 35, and 41.)

[SLG01] William Su, Sung-Ju Lee, and Mario Gerla. Mobility prediction and routing in ad hoc wireless networks. *International Journal of Networking Management*, 11(1):3–30, 2001. (Cited on page 23.)

[SOH02] Mansorr Shafi, Shigeaki Ogose, and Takeshi Hattori. *Wireless Communications in the 21st Century*. Wiley-IEEE, 2002. (Cited on page 93.)

[Sol96] J. Solomon. Applicability Statement for IP Mobility Support. RFC 2005 (Proposed Standard), October 1996. (Cited on page 5.)

[SPR04] A. Spuropoulos, K. Psounis, and C. Raghavendra. Single-copy routing in intermittently connected mobile networks. In *Proceedings of IEEE SECON*, October 2004. (Cited on page 110.)

[SPR05] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proceedings of SIGCOMM 2005*, 2005. (Cited on page 110.)

[SXM$^+$00] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. RFC 2960 (Proposed Standard), October 2000. Updated by RFC 3309. (Cited on page 14.)

[Tab00] Sami Tabbane. *Handbook of Mobile Radio Networks*. Artech House, 2000. (Cited on page 2.)

[TBA01] Anup Kumar Talukdar, B. R. Badrinath, and Arup Acharya. MRSVP: a resource reservation protocol for an integrated services network with mobile hosts. *Wireless Networks*, 7(1):5–19, 2001. (Cited on page 2.)

[TKCK02] Dirk Trossen, Govind Krishnamurthi, Hemant Chaskar, and James Kempf. Issues in candidate access router discovery for seamless IP-level handoffs. Draft, October 2002. `draft-ietf-seamoby-cardiscovery-issues-04.txt`. (Cited on page 13.)

[TLLW03] Chien-Chao Tseng, Gwo-Chuan Lee, Ren-Shiou Liu, and Tsan-Pin Wang. HMRSVP: a hierarchical mobile RSVP protocol. *Wireless Networks*, 9(2):95–102, 2003. (Cited on page 2.)

[TN98] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. RFC 2462 (Draft Standard), December 1998. (Cited on page 25.)

[Tol99] V. Tolety. Load reduction in Ad Hoc networks using mobile servers. Master's thesis, Colorado School of Mines, 1999. (Cited on page 96.)

[TR01] Fabrice Tchakountio and Ram Ramanathan. Tracking highly mobile endpoints. In *WOWMOM '01: Proceedings of the 4th ACM international workshop on Wireless mobile multimedia*, pages 83–94, New York, NY, USA, 2001. ACM Press. (Cited on page 110.)

[TRV98] N. Tripathi, J. Reed, and H. Vanlandingham. Handoff in cellular systems. *IEEE Personal Communications*, 5(6):26–37, December 1998. (Cited on page 3.)

[TYK05] Adam Theissa, David C. Yena, and Cheng-Yuan Ku. Global Positioning Systems: an analysis of applications, current development and future implementations. *Computer Standards & Interfaces*, 27, June 2005. (Cited on pages 7 and 23.)

[TZZ03] K. Tan, Q. Zhang, and W. Zhu. Shortest path routing in partially connected Ad Hoc networks. In *Proc. of IEEE GLOBECOM'03*, volume 2, pages 1038–1042, December 2003. (Cited on pages 110 and 121.)

[Val99] András G. Valkó. Cellular IP: A new approach to Internet host mobility. *Computer Communication Review*, 29(1):50–65, January 1999. (Cited on page 6.)

[VB00] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report TR CS-200006, Duke University, April 2000. (Cited on page 110.)

[vdL97] Jan van der Lubbe. *Information theory.* Cambridge University Press, 1997. (Cited on pages 18 and 72.)

[WJMF05] Yong Wang, Sushant Jain, Margaret Martonosi, and Kevin Fall. Erasure-coding based routing for opportunistic networks. In *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 229–236, New York, NY, USA, 2005. ACM Press. (Cited on page 110.)

[WKMT00] David H. Wolpert, Sergery Kirshner, Chris J. Merz, and Kagan Tumer. Adaptivity in agent-based routing for data networks. In *Proc. of the fourth international conference on Autonomous agents (AGENTS'00)*, pages 396–403, New York, NY, USA, 2000. ACM Press. (Cited on page 21.)

[YAG05] L. Yann-Ael and B. Gianluca. Round robin cycle for predictions in wireless sensor networks. In *Proc. on Intelligent Sensors, Sensor Networks and Information Processing Conference*, pages 253–258, December 2005. (Cited on page 21.)

[YHP02] Jian Ye, Jiongkuan Hou, and Symeon Papavassiliou. A comprehensive resource management framework for next generation wireless networks. *IEEE Transactions on Mobile Computing*, 1(4):249–264, 2002. (Cited on page 92.)

[YKUM05] Gokhan Yavas, Dimitrios Katsaros, Ozgur Ulusoy, and Yannis Manolopoulos. A data mining approach for location prediction in mobile environments. *Data & Knowledge Engineering*, 54(2):121–146, 2005. (Cited on pages 21 and 23.)

[YL02] Fei Yu and Victor Leung. Mobility-based predictive call admission control and bandwidth reservation in wireless cellular networks. *Computer Networks*, 38(5):577–589, 2002. (Cited on pages 17, 20, and 35.)

[ZBRC06] Sofia Zaidenberg, Oliver Brdiczka, Patrick Reignier, and James L. Crowley. Learning context models for the recognition of scenarios. In *3rd Conference on Artificial Intelligence Applications & Innovations (AIAI) 2006*, volume 204/2006 of *International Federation for Information Processing*, pages 86–97. IFIP, Springer Boston, June 2006. (Cited on pages 53 and 135.)

[ZCCS02] Z. Zhou, R. Chen, P. Chumchu, and A. Seneviratne. A software based indoor relative location management system. In *Proceedings of Wireless and Optical Communications*, Canada, 2002. (Cited on page 16.)

[ZD97] Mahmood Zonoozi and Prem Dassanayake. User mobility modeling and characterization of mobility patterns. *IEEE Journal on selected areas in communications*, 15(7):1239–1252, 1997. (Cited on page 96.)

[Zha06] Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay toerant networks: overview and challenges. In *IEEE Communications Surveys and Tutorials*, volume 8-1, pages 24–37, 2006. (Cited on pages 109 and 111.)

[Zim80] Hubert Zimmermann. OSI reference model—The ISO model of architecture for Open Systems Interconnection. *IEEE Transactions on communications*, 28(4), April 1980. Invited paper. (Cited on page 3.)

[ZL78] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978. (Cited on page 19.)

[ZL01] Y. Zhang and D. Liu. An adaptive algorithm for call admission control in wireless networks. In *Proc. of the IEEE Global Communications Conference*, pages 3628–3632, San Antonio, TX, November 2001. (Cited on page 93.)

[ZM05] Zainab R. Zaidi and Brian L. Mark. Real-time mobility tracking algorithms for cellular networks based on Kalman filtering. *IEEE Transactions on Mobile Computing*, 4(2):195–208, 2005. (Cited on page 23.)