Rapport de première année de thèse

Réseaux diffus et applications pour l'informatique ambiante

Olivier Mehani*

Centre de Robotique
École des Mines de Paris

*olivier.mehani@ensmp.fr

Résumé

Les équipements informatiques sont de plus en plus petits, ce qui a pour conséquence une plus grande portabilité, et une plus grande facilité d'embarquement dans des objets de tous les jours. Avec la généralisation des réseaux sans fils, ces appareils sont maintenant souvent équipés d'une ou plusieurs interfaces réseau leur permettant de communiquer les uns avec les autres.

Cette situation rend possible de toutes nouvelles applications. Il n'est plus impossible d'imaginer un ensemble d'appareils travaillant ensemble afin d'accomplir une tâche complexe, commandée par l'utilisateur, que chacun d'entre eux n'aurait pas été capable d'accomplir seul. De telles tâches peuvent varier de trouver une route vers l'Internet à établir la combinaison d'équipements la plus adaptée pour restituer du contenu multimédia ou des informations à l'utilisateur. Il est en fait possible d'imaginer des appareils robotisés, complètement autonomes et communiquants, coordonnant leurs actions afin de réaliser un objectif de haut niveau.

L'architecture réseau nécessaire pour les applications présentées ici, bien que basée sur les même technologies sous-jacentes, n'a que peu en commun avec la structure habituelle. Ces nouveaux types de réseaux sont en général appelés "ubiquitaires", "diffus" ou "ambiants", pour mettre l'accent sur leur présence discrète, mais disponibilité permanente.

De nouvelles conditions d'utilisation, cependant, créent de nouveaux problèmes qui doivent être résolus afin d'assurer le fonctionnement désiré du système. Nous proposons, dans cette étude, d'identifier ces problèmes liés réseaux diffus, et d'y apporter, après évaluation, des solutions efficaces.

Abstract

Computing devices are becoming smaller and smaller and, as a result, more and more portable or embedded into everyday objects. With the generalization of wireless networks, these devices now boast one or more network interfaces allowing them to communicate with one another.

This situation leaves room for new applications. It is no longer impossible to imagine a set of devices cooperating with each other in order to accomplish a complex task, demanded by the user, which any of them would not have been able to achieve on its own. Such tasks vary from finding a route to the Internet to building the most well suited combination of equipments to deliver multimedia content or information to the user. One can even think of completely automated robotic devices communicating with each other and coordinating their actions to achieve a higher level goal.

The network architecture needed for the applications presented here, even though it is based on the same underlying technologies, do not have much in common with the usual structure. These new networks are usually referred to as “ubiquitous”, “pervasive” or “ambient”, to emphasize on their discreet presence, but permanent availability.

New conditions of use, however, create new issues that have to be addressed to enable proper operation of the system. We propose, in this study, to identify these pervasive networking problems and to offer and evaluate solutions to overcome them.

Contents

Chapter 1

Introduction

With the growing number of devices used either in fixed or mobile environments (PCs, laptops, PDAs, phones, etc.), the concept of Ambient Intelligence, which exploits the available appliances and networks within a context of current "ambient", *i.e.* user explicitly stated or derived, preferences, situation, etc., becomes increasingly interesting. Additionally, one can also further exploit the convergence of all the computing terminals into one single unit which would be able to provide all the current usual services (telephone, text messages, etc.) as well as new applications made possible by an ubiquitous network connectivity to both local and remote services providers.

Challenges towards the full realization of this goal lie in the fact that the current network infrastructure is based on heavy assumptions which no longer hold in when networks become "pervasive". The biggest issue is related to the fact that devices become highly mobile, but are still expected to operate as if fixed (continuity of connection, reachability,...). As a side effect of this high mobility, it is no longer possible to hard-code network parameters (such as network address or a list of usual service providers), as they would perpetually change, depending on where the device is and what is in its vicinity. Some new ways to adapt a node's configuration then have to be identified and developed.

This report focuses on the work done during the first five months of PhD research, under a cotutelle agreement with UNSW[1], at Nicta[2] in the *Networked Systems Theme*, as well as some relevant previous works which lead to the submission of a research proposal on "Pervasive Networks and Ambient Intelligence Applications".

The rest of this document is organized as follows. Chapter 2 refines the problem statement and gives some basic use cases in the form of scenarii. Chapter 3 presents a preliminary survey of the relevant literature while chapter 4 puts an emphasis on what has not been addressed yet. Finally chapters 5 and 6 respectively present the work done so far, and the sketch of a programme for the upcoming months.

[1]The University of New South Wales, http://www.unsw.edu.au
[2]National ICT Australia, http://www.nicta.com.au

Chapter 2

Problem Statement

2.1 Requirements

Ubiquitous connectivity and seamless access to services has long been a goal for researchers. With the increasing deployment of wired and wireless networks and means of accessing available services, this goal is getting closer to reality. However, more connectivity choices present additional challenges with respect to increased usability, quality of service and means of optimizing the access method so that the user, network operator or service provider preferences can be considered and met. A number of issues can be identified in these areas.

Automatic discovery of neighboring systems and services is a key enabling feature for pervasive networking. It is indeed highly desirable to be able to detect the local context (*e.g.* devices in the vicinity), or search for specific services. Such a system should act in a way to provide an always up-to-date map of the ambient surroundings, available networks and services providers.

The problems of mobility, seamless access method changes and usage of the best available resources (eventually minimizing this usage by only transferring meaningful information) are the basic issues to solve for any communication. Switching from a home environment, with a high bandwidth uplink, to a less equipped place (*e.g.* a street with wireless connectivity) should have the least noticeable impact on the available services and their quality.

Addressing these pervasive nodes in the network may become an issue. The current addressing scheme indeed relies on the assumption that the network infrastructure is highly hierarchized and somehow manually configured by network administrators. Although it applies well to regular networks, this supposition does not hold in a highly dynamic network where devices may opportunistically change their access router, or not have one at all, while still having to communicate with other nodes.

It is also critical, in a context where more and more devices try to use the available connectivity, that the protocols used to carry data compete fairly for bandwidth that is, adapt their demands to what the network can cope with without starving other nodes. Congestion control algorithms have been devised with fairness to others in mind and already implemented in some of the regular transport protocols for decades. It becomes, however, increasingly important that *all* transports used in a pervasive network environment are able to regulate their own bandwidth consumption.

2.2 Scenarii

Several scenarii can be derived in order to demonstrate some typical (unrefined yet) use cases where the above requirements come into play.

2.2.1 Smart Home

The typical use case of pervasive *computing* is the smart home, where devices discreetly cooperate with each other to give the best usage of the technology.

A user terminal should be able to use the pervasive home network to browse the ambient services provided by local devices. Once discovered, it should also be possible to establish network link between services, so as to move data from where it stands to where it is useful.

Hence, starting a video conference with another person would exploit the local devices, like a flat screen or a stereo system, by sending them the appropriate media streams as they come through the home Internet gateway. If the user subsequently moves while still in communication, said devices or even the Internet gateway may become unavailable (*e.g.* the user is now walking in the street). The needed data streams and output media to keep the communication going would then have to be transparently relocated and adapted (*e.g.* using a 3G interface, canceling the video stream and moving the audio stream, using a less network-intensive codec, to a headset).

2.2.2 Road Automation

Road vehicles are sporting more and more electronic devices to get information about the local state or the traffic conditions. This information can be of great help for increasing road awareness and safety.

Embedding a pervasive networking device in these vehicle can allow for useful exchange of valuable information. One example can be two cars crossing one another in opposite direction, and exchanging recent traffic information about where they came from with information about where they are going. Another interesting usage could be synchronization of digitized maps if one car happens to have been to a place which geography has recently changed (*e.g.* new road setup or driving rules), or simply to have had its map updated more recently.

As an addition to the smart home scenario, such a vehicle can also be seen as an extension of the house. End-user devices should be able to interact with the car's inner system (*e.g.* retrieve and display information about the journey or the vitals of the vehicle), user devices in other vehicles (*e.g.* establishing and maintaining a voice conversation with a person aboard another car) or the global network (*e.g.* accessing the Internet via the car's uplink, be it direct or not).

Chapter 3

State of the Art

3.1 Discovery of Locally Available Services

Services discovery in a network are an increasingly desirable, and researched, functionality for configuration-less devices. Pervasive computing environments have a larger set of requirements which already existent SD systems fulfill with a variable success rate [1].

The following presents a short, still incomplete, survey of some of the most prominent service discovery proposals which could be successfully adapted to an ubiquitous network environment.

3.1.1 Generic Services Discovery Systems

A Service Location Protocol has been proposed as early as in 1997 and, subsequently, a second version [2, 3] was introduced in 1999. It supports resolving a type of service, more precisely identified using a set of attributes, into the network address of a node providing the required functionality. SLP is a request-based scheme and supports two modes of operation: direct or directory-based. In direct mode, a *User Agent* sends its request to a specific multicast group to which *Service Agents* are registered. Service Agents providing a matching service give the information back to the client using unicast replies. Directory-based mode can be used for larger networks, in which a User Agent sends an unicast request to a centralized *Directory Agent*, to which the Service Agent have previously given information about what they were able to provide. The Directory Agents are located by the User Agents using a multicast request, or periodically sent advertisements. The use of different *scopes* can be used if more granularity in the available services is needed (UA will only discover services from SA within the same scope).

More recently, Apple (which previously used SLP) introduced a DNS-based Service Discovery method [4]. This relies on the `SRV` record type and two subdomains, `_tcp` and `_udp`, to store pointers to services available for a specific domain. Being based on the highly centralized DNS infrastructure, this scheme by itself is not well suited for the possibly disconnected pervasive subnetworks that may be created sometimes. To allow for decentralized operation, Multicast DNS [5] can be used. This mechanism allows every node in a cluster to send DNS (including DNS-SD) requests to a multicast group, to be answered by local stations having relevant information. A cache is maintained in every node to reduce the network impact. In addition to a request-based scheme, nodes which network connectivity or status change briefly update their neighbors' cache by sending advertisement about all their hosted services.

The Simple Service Discovery Protocol [6] (on which the Microsoft-backed Universal Plug'n Play (UPnP) system is based), proposes an extension to the HTTP protocol, to be used over UDP on a multicast channel, to search close-by devices for the desired service. It introduces a new naming convention, by attributing specific URIs to services and nodes (identified by their UUIDs). Support for service presence announcements allows service providers to inform the stations about their status change, which (as for the advertisement scheme in mDNS) can reduce the amount of requests client node would subsequently issue.

3.1.2 Ad-hoc and Mesh Networks Extensions

Several improvements have been proposed for some of these protocols to perform better in pervasive networks. More specifically, they focused on mesh networks (and their subsets: ad-hoc networks), which are supposed to be a typical interconnection scheme of pervasive networking devices.

An adaptation of SLP to ad-hoc networks, SLPManet, was proposed in [7]. It first removes some of the optional features of SLP which do not adapt well to MANET (*e.g.* centralized Directory Agents). Instead, it relies on nodes in a mesh network forwarding an SLP Service Request only if they do not have information about such a service in their cache, or directly answering the request with the cached information otherwise. This is suboptimal as it could conceal some available services: a node knowing about only one of two available services does not propagate the request, and imcomplete replies are sent back to the querying node. To overcome this issue, SLPManet proposes an *extended caching* optimization, in which the cache is used only to answer queries from applications running locally (*i.e.* on the same node), still forwarding the requests from other nodes. However this mitigates the issue of having an incomplete view of the surrounding services, some devices, particularly the ones that were involved in forwarding previous searches results for similar services, only use the fragmentary information from their cache. All in all, it is unsure whether the savings in terms of bandwidth are worth the incompleteness of the information that this creates.

An interesting cross-layer approach to give better performances for mDNS-SD has been presented in [8]. It supposes that the mesh network's network routes are maintained using the Optimized Link State Routing (OLSR) protocol [9]. The main goal of this approach is to use the routing features of OLSR to adequately deliver the DNS-SD queries to the whole network. Instead of encapsulating DNS-SD queries into mDNS messages, they are given an OLSR header. The presented implementation keeps the mDNS caches of the relaying nodes up-to-date by extracting the information before re-forwarding the OLSR packet. Additionally, intermediate routers are allowed to give non-authoritative (but identified as so) answers to queries for which cache-hits happen. An interesting example of cross layer design, this method tries to compensate for the unavailability of multicast routing within an OLSR network. This solution may however soon be outdated as some works [10] propose solution to relay multicast traffic throughout such a system.

3.2 Mobility at Different Layers

Mobility handling mechanisms have been proposed at almost all layers from link to application of the OSI model [11]. Even though some arguments suggest that these should lie in one layer rather than the other of the OSI stack [12], the question remains largely open.

3.2.1 Link Layer

Link layer mobility is really dependent on the physical link technology. Depending on whether the old link and the new link are (resp.) part or not of the same network, link-layers handoffs can be (resp.) transparent or not to the upper levels. In the latter case, additional mobility support in the layers on top is required. Most of the proposed protocols at this level are designed to speed up handovers.

3.2.2 Network Layer

Network layer mobility solutions like MIP(v6) [13, 14] or NEMO [15] allow a node (or network) to maintain connections between a specific network address (its *home address* or *HoA*) and the rest of the network. The node keeps its HoA even though it no longer has direct connectivity to its home network using tunneling and encapsulation. This mobility scheme requires a modification in the infrastructure in the form of a *Home Agent* which would take car of en-/decapsulating and forwarding packets for the *Mobile Node* (or Network). On the *Correspondent Nodes*' part, no modification is actually required, though communications can be optimized if they are MIP-aware.

No further modification is required in the applications or the rest of the network. This mobility scheme is completely transparent for the higher levels, but introduces a delay while the network topologically-correct address changes. These are the sum of several factors including the layer 2 handoff, the time to configure the new *Care-of Address* (CoA) after a *Router Advertisement* from the new network has been received, and the exchange of *Binding Updates* and acknowledgment with the HA. These delays can impair the proper working of the transport layers.

As a slightly different approach, MIP-LR [16] suggests the use of distributed *Location Registers* in place of the regular Home Agent to mitigate single-point-of-failure issues (mostly in a military context). No encapsulation is performed, removing the tunneling overhead as well as the triangular routing issue: correspondent nodes query the HLR for the current network address of the MN they want to contact, then establish the connection directly using the obtained CoA. This scheme, however, no longer has the benefit of transparently interoperating with legacy IP hosts.

LIN6 (Location Independent Networking for IPv6) [17] proposes the same type of approach based on *Generalized IDs* which are valid IPv6 address uniquely identifying a host. A *Mapping Agent* is associated to everyone of these identifiers, and can be found using the DNS system. Unlike MIPv6, LIN6 does not rely on tunneling, but rather on live replacement (*embedment*) of the generalized ID by the current locator of the MH, taken from the sending node's mapping table. Whenever the MH moves, it notifies its Mapping Agent which in turn informs the Correspondent Nodes. This scheme is lighter on the network impact, as no encapsulation is done, but needs modification of the communication stacks of both peers to add support for embedment of the generalized IDs.

HIP (Host Identity Protocol) [18] works on a similar concept except that it introduces a new 128 bits hashed identifier used to decouple the network and transport layers. The identifier being different from regular IPv6 addresses, this solution needs the upper level applications to be modified in order to benefit from it.

3.2.3 Transport Layer

Transport layer mobility allows to dynamically change the endpoint of a connection to another network address, without need for the upper layers to support or even be notified of the changes.

A suggested extension to the DCCP protocol [19, 20] (see section 3.3 on page 12) is the generalized connections mechanism [21]. In this scheme, a mobile host can add or remove components (*i.e.* network endpoints address) to/from an already established socket without interruption of the communication.

MSOCKS [22] proposes a transport layer proxy between the mobile node and its correspondent node. Every connection is given an identifier which the MN can use, after a change of network address, to re-establish the link, and resume the data exchange.

Several additional extensions to TCP have been proposed as well. Two main types are noticed. The first ones are based on some sort of secret (sometimes called *cookie*) exchanged at connection-establishment time in order to be able to authenticate future reconnection from new network addresses: TCP Migrate [23], TCP-R [24], TCP Mobility [25],... The second type, such as I-TCP [26], relies on remote agents splitting the connection in halves and abstracting the mobility of one peer to the other, much in the way MSOCKS does.

SCTP [27] (see section 3.3) has the ability to register new network addresses for use with an already established connection. The MH can then, after having moved to the new network, send a request to resume the communication from there.

3.2.4 Session Layer

Session layer mobility alleviates the cost of broken connections at the lower levels by keeping state information and re-establishing lost connections to the peers.

Instead of establishing a direct connection between hosts, SLM [28] introduces the concept of *User Location Server* which keeps track of the mobile hosts' current network address. An application desiring to establish a connection will contact its *reflector* which will search the DNS

entries for the desired node's ULS in order to obtain its current network address. The reflector then establishes a connection to the Mobile Host's *connector* at the resolved address. The connector completes the link by connecting to the local application on the MH. Whenever the MH changes its point of attachment to the network, neither part of the connection can notice the change as they are both locally connected to the reflector or the connector, which take care of re-establishing the connection to the new MH's address.

This solution is not straight-forward, though, as it requires both to deploy new DNS entries, and to slightly modify the applications (or the underlying communication libraries) to connect via the reflector instead of directly to the node.

3.2.5 Application Layer

Application layer mobility is usually an ad-hoc per-application or per-protocol solution to compensate, or prevent, connection cuts on the lower layers.

L-7 Mobility [29] proposes the use of a local proxy. The connecting application (on the mobile host) would connect to this proxy which in turn would identify the proper way to connect (be it directly, through another proxy, etc.) with the peer. As mobility is handled at layer 7, with insight into the application's protocol, there should exist one type of proxy per application protocol. This may not be suitable, though, for all types of protocols as the proposed solution needs idempotent requests. A request shall, indeed, be issued (and processed at the receiving node) more than once, when handoffs occur, in order to deliver a complete answer to the client application. Moreover, this solution only allows mobile nodes to initiate the connection.

SIP, and more precisely ALM-SIP [30], do not rely on the network addresses to find MNs but rather on an identifier – very similar to an email address: `ID@DOMAIN` – which identifies the application (or user) and a way to contact it. This protocol relies on a number of SIP gateways (similar to HTTP proxies) to relay messages and flows to the destination. Every SIP client keeps its originating domain's registrar updated with information about how it can be reached at the time, so that any application trying to find the client can be properly redirected. SIP provides signaling and content negotiation between clients and can encapsulate any type of stream (feature on which ALM-SIP is based).

SIP has the additional property to handle service mobility over devices, rather than just that of the devices over networks. In any case, application layer mobility needs the existing applications to be modified in order to get benefits from the mechanism.

3.2.6 Multihoming

In addition to plain mobility, several of these protocols have support for multihoming (*i.e.* one MH has several concurrent points of attachment to the network topology, hence several network addresses).

On the network level, the Multiple Care-of Address [31] extension allows a MIPv6 node to register several bindings with its HA. HIP has support, through its `LOCATOR` parameter, to use several network addresses as well.

Several transport protocols also support multihoming. SCTP natively supports multiple transmission paths, whereas the TCP-MH [32] options allow a host to associate more than one network address to a socket. Similarly, the handling of more than one network address is an obvious feature of DCCP's generalized connections mechanism. Finally, at the session/application level, SIP [33, 34] can direct all or parts of the streams of a single session to different network addresses (which may not even belong to the same host).

3.2.7 Additional MIPv6-related extensions

Handovers Delay-Mitigation Techniques

As mentioned earlier, handoffs cause delays which may alter the performances of other layers. Some have been proposed to make MIPv6 handovers faster. These usually rely on getting information

from the link layer [35] before the handoff is performed.

Fast Handovers for Mobile IPv6 [36, 37] focuses on reducing the delay induced by layer 2 handovers. The mobile host, once it has established connectivity with an access router, keeps sensing other (typically wireless) networks and requests information about their subnet configuration via its current *Access Router*. This prevents the node from waiting for router advertisements to configure its new CoA after having connected to a new network: the node can use the previously determined CoA for the network immediately. Before disconnecting from its previous access router, the MN can also send a message to the AR informing it about the next network it will visit. The previous AR can then establish a tunnel to the new AR. Packets destined to the old CoA are then forwarded via this tunnel to the new CoA until the Binding Update process has been completed with the HA.

Recently, works have been oriented towards an abstraction of layer 2 states and events [38] to allow for less dependence of the mobility subsystem on the type of link layer. With this level of information, the layer 3 mobility algorithms can prepare the handover with other ARs in advance, and instruct the layer 2 to hand off at the most appropriate time (before the signal would have completely faded) [39].

Micro-Mobility

In order to yield better performances and reduce the global network traffic footprint caused by a node moving within the same administrative domain, various *IP micro-mobility* [40, 41] schemes have also been proposed[1]. Their role is to mitigate the impact of a node roaming between several subnetworks of a unique *Autonomous System* (several IP networks under the control of one entity) by having its local network address unchanged. No CoA update then need to be propagated to the HA, and routing and/or tunneling of the packets throughout the AS is taken care of by the local infrastructure.

Hierarchical Mobile IPv6 (HMIPv6) [42] is based on the presence of *Mobility Anchor Point* in the domain, redirecting traffic to the MH's RCoA (*Regional CoA*), which would be the CoA registered to its HA, to its *Local CoA* (LCoA), the node's actual topologically address at the moment. On any subnetwork change, the HMIPv6-compliant MN learns its new LCoA via the Router Advertisement (with a specific MAP option), and sends *Local Binding Updates* to the MAP for it to properly route traffic received for the RCoA.

MIPv6 Neighborhood Routing for Fast Handoff [43] proposes an extension to allow an MN to register both its current CoA and the ones it would have in other subnetworks. Encapsulated packets for the MH are first sent to its current CoA with a specific routing header instructing that they should be forwarded, by the local routers, to the next subnetworks if the node is no longer reachable at the specified CoA at the time of delivery.

3.3 Congestion Control to Regulate the Network Load

Currently used flavors of TCP include efficient congestion control mechanisms [44] which allow the hosts to automatically adapt their streams' speeds in order not to saturate the network paths and be fair to other traffic following similar routes. These stacks implement one or more control algorithms, generally based on the *Additive Increase/Multiplicative Decrease* (AIMD) feedback control algorithm with additional optimizations. Even though some issues may appear in the case of wireless networks where packets losses may not be related to congestion, for which enhancements can still be brought (as in, *e.g.* [45]), this field is fairly well covered.

UDP, the historical non-reliable datagram-oriented transport protocol, however, does not provide any congestion detection mechanism. This is mostly due to the fact that no acknowledgment is sent back to the sender. Thus, it cannot estimate current status of the network. Several more recent transport protocols try to address this deficiency by allowing for congestion-controlled datagram-based communication.

[1] both for MIPv6 and other architectures, though focus here is on the former only.

The Stream Control Transmission Protocol [27] can embed several of said streams in one single connection. Congestion control is achieved in a scheme much similar to that of TCP, including Slow Start and Congestion Avoidance modes based on a Congestion Window of variable size. SCTP also ensures that the datagrams are reliably delivered to the receiver.

The Datagram Congestion Control Protocol [19, 20] has been proposed to provide congestion control *only* for datagrams, without enforcing delivery reliability (*i.e.* no reordering upon arrival, and no retransmission of lost datagrams). It has been designed with modularity in mind, and several congestion control mechanisms (identified by their CCID: Congestion Control IDentifier) can be chosen.

A TCP-like congestion control algorithm, CCID2 [46], mimics the behavior of TCP's Congestion-Window-based mechanism, including Slow Start and Congestion Avoidance Modes.

CCID3 [47] proposes the use of the TCP-Friendly Rate Control algorithm [48]. TFRC is an equation-based sending rate control mechanism. It uses network-gathered metrics like the *Round Trip Time* (RTT) or the number of packet losses (more precisely, the size of the *loss events*) to compute the allowed rate. The equation used for DCCP is a simplified version of TCP's throughput model (Eq. 3.1, where s is the packet size, p the loss event rate, R the measured RTT and t_{RTO} the retransmit timeout). CCID4 [49] as also been proposed as a variant of CCID3 for small packets.

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p\left(1+32p^2\right)} \tag{3.1}$$

Both SCTP and DCCP are proposed to carry media streams. DCCP, however, stays closer to UDP semantics by not enforcing reliability on the datagrams delivery. Additionally, the use of an equation-based rate control (in the case of CCID3) makes the rate changes smoother, which is more appropriate for the streaming of multimedia content. Together with the proposed CCID4, the non-reliability of DCCP allows for a more timely delivery of datagrams, which is highly desirable in the case of interactive media traffic, like VoIP of video conferencing.

Chapter 4

Identified Open Issues

4.1 Service Discovery for Pervasive Networking

Most service discovery protocols are not designed with mobility in mind. The first objective is to alleviate the need for costly (in terms of resources, personnel or skills) configuration of devices to use with local services such as a DNS server or a printer. Though most of them can handle the occasional movement of a node from one network to the other, they do not accommodate well to fast moving stations.

In the light of pervasive computing, the requirements for a well-suited service discovery system are much stronger. A service discovery solution for ambient networking which supports the following needs still has to be extended or designed.

- decentralized operation;
- peer-to-peer (ad-hoc) operation;
- cross-technology operation, at least at the physical layer;
- support for rapid movements of clients (and services) between networks.
- sufficient expressivity of the properties so that an appropriate choice from several similar services can be made.

4.2 Two Different Needs for Mobility

With respect to the literature, it turns out that "mobility" covers two slightly different fields. Depending on what entity (be it a physical device or a user session) moves, mobility would not be best addressed at the same layer.

Host mobility occurs when network streams remain tied to a single host which moves across the network and changes its topological address. In this case, it is necessary to have an identifier for the host that is independent of its network topological address, now subject to change. Several identifiers have been proposed, frequently adding virtual layers in the stack, but the MIPv6 [14] approach has this advantage that it still and only (*e.g.* no use of the DNS infrastructure) relies on topologically correct network addresses, implying that the higher levels (and correspondent nodes in a first approach) do not have to be modified. In this respect, address-based multihoming would also be (almost) transparently addressed at this layer.

Session mobility, on the other hand, would deal with all or part of the network streams composing a session explicitly changing their point of attachment, usually moving to another device. This scheme would usually require prior negotiation between the current destination host and the future one in order to transmit session states, then an express move message to the sending host so that it redirects its stream(s) to the new device, or starts a content negotiation phase in order to get the

best out of the new host's (and network path) offered capabilities. This kind of mobility, having to be well aware of the data that are dealt with, would be better handled at higher levels that is, the application layer or, better, the session layer. Contrary to host mobility, this is not transparent at all to applications, as they have to go through a layer that was previously unimplemented. This is not, however, as problematic as it would sound as applications taking advantage of this kind of mobility are not widely deployed. Moreover, one could imagine session-mobility-enabled hosts transparently communicating with mobility-disabled nodes thanks to the session-state exchange that took place beforehand (e.g. a file transfer using FTP could be resumed from another host knowing where to restart the download, without any modification on the serving part). In this scheme, sessions also need to be uniquely identified. SIP, however not exactly at the session level, provides a rather complete framework for stream mobility inside a session, as well as sessions and application endpoints identification. It then seems to be a good basis to be extended to address more session mobility scenarii.

4.3 Reducing the Impact of Mobility on Performances

Congestion control algorithms implemented as part of the transport protocols are a desirable feature. These protocols usually rely on the detection of packet losses. In a wired system with static nodes, packet losses are due to routers' buffers being full. Hence, a packet loss is the symptom of a congested network path. In mobility situations, though, packet losses can be due to other causes, and result in unneeded congestion avoidance mechanisms being triggered.

Considering host mobility at the network layer, some modifications could significantly improve the behavior of the data streams. With the possibility of cross-layer optimization, the impact of handoffs on the congestion control algorithms, or even the overall system, could be greatly attenuated.

Using information from both lowest layers at the network level has already been proposed to achieve faster handoffs [35]. Similarly, sharing information from the lower level with the transport layer can help mitigate the packet losses due to temporary physical or logical disconnections [45].

Extensions to MIPv6 like NEMO [15] can add complexity to these cross-layer mobility handling mechanisms as the Mobile Network Node (MNN) is not directly responsible (or even aware) of a handover that about to be performed. As the Mobile Router would mostly be, in this case, a device without direct interaction with the user, it is not acceptable to only improve its own connections. Indeed, an user's tasks and connections are more important. Mechanisms for them to benefit of such optimizations need to be devised.

4.4 Proposed Approach

In order to build a well suited protocols and applications stack for pervasive networking, the aforementioned issues have to be addressed in an adequate way. We suggest a bottom up approach.

For a start, the malfunctions caused to the transports by host mobility situations are considered. Extensions allowing them to differentiate the reason of, or avoid if possible, packet losses shall be proposed, implemented and evaluated. Generic methods to exchange information between layers, both within the same node or between stations (*e.g.* as may be required for NEMO) will also be introduced.

Contextual information about the pervasive computing environment can be of use both to optimize the operation of the lower layers as well as, of course, to enable applications to be aware of the others' availability and the services they can provide. A way to gather information about what tasks the devices in the vicinity can do on behalf of the applications shall be studied.

Eventually, a framework integrating the previous functionalities to build pervasive networking applications is to be researched and developed.

Chapter 5

Past and Current Works

5.1 Service Discovery and Mobility in VANETs

Nota: this section quickly summarizes work that has been started before the beginning of this thesis, but covers questions tightly related to the topic and should be covered more in depth shortly.

A Vehicular Ad-hoc NETwork (VANET) is a challenging pervasive networking environment. This is mainly due to the fact that this is a highly dynamic network in which two nodes can as quickly come in the vicinity of one another as they can move away.

When these vehicles, which may embed more than one computing device, are getting closer, it is important to have reactive enough a discovery mechanism so that entities in these vehicles (both automatic programs or human beings) can assert whether the other vehicle can fulfill one or several networking or information-related need. If this is the case, usual network connections can be established in order to complete the desired task. Due to the differences in velocity or journey of the vehicles, direct communication may not be sustainable for more than tens of seconds. Routing solutions can be integrated using other vehicles or Internet uplinks to relay traffic until the connection is no longer needed.

5.1.1 Implementation of mDNS-SD

To allow applications embedded in vehicles in geographical proximity to cooperate, it is first needed to have dynamic and decentralized means to detect their peers. The mDNS-SD protocol has been integrated into vehicular routers [50].

This technology relies on small multicast messages to advertise or request services. Experiments have shown a very reasonable network usage, of the order of an average of 5 Bps per station, with peaks at 30 Bps after a network topology change, which has been considered small enough to be acceptable in such a network environment.

As the multicast group in use falls in a non-forwardable address range, aggregation and replication of the information was used to get it to the end-user devices in the local subnetworks of the vehicles. However this scheme performed well in static or low speed scenarii, latencies (of the order of 5 s) where observed, which may result in the impossibility to accurately detect close-by devices if the dynamicity of the network increased.

Additionally, only basic network services were discoverable, which may not be sufficient for an application to make choices on behalf of its user. To provide for more expressivity of the services' properties, an ontonlogy server was additionally used, and advertised.

A close approach, integrating an ontology (or a similarly expressive system) more tightly into the basic discovery scheme as well as increasing the reactivity, may be a good basis for a specifically crafted service discovery architecture for ambient networking.

5.1.2 Network Mobility in a VANET

In [51], route optimization and multihoming have been used to provide an efficient, and uninterrupted connectivity between moving vehicles. They were equipped with several network interfaces and, depending on whether the vehicles were in local proximity, close to a relay network, or too distant to establish a direct connection, several access technologies (ad-hoc or managed Wi-Fi and GPRS) were used.

Network mobility for IPv6 was used to abstract which interface was in use from the applications. Some information about the type of connection in use was, however, available to the applications to adapt the format data they were exchanging.

This work showed the feasibility of such an infrastructure. The concepts validated for this sort of architecture now have to be generalized and better integrated.

5.2 Influence of Mobility on Transport Protocols Efficiency

5.2.1 Packet Losses Due to Handovers Rather Than Congestion

In mobility situations, in which a node (wired, wireless or both) moves from one point of attachment to another in the network, there exists a delay caused by the handover process. During this period, the node is completely unreachable, and even if the current active connections do not timeout, a significant number of packets may be lost in the process.

In this situation, packet losses are obviously not a result of congestion. The congestion control algorithms, though, do not have any means of differentiating one case from the other, and always react as if the network path were overloaded. The usual congestion-mitigating behavior consists in reducing the number (and/or size) of packets sent over time. This can range from a division of the current sending rate to a reset to the lowest achievable one. This unnecessarily reduces the performances of the transport protocol if a packets have not been lost due to a congested network.

5.2.2 Simulated Impact of Handoffs on Transport Protocols

The impact of mobility-caused temporary disconnections has been studied in simulation for both TCP and UDP, as well as DCCP.

Simulations have been run in the NS-2 network simulator[1] version 2.32. Mobility models only exist for IPv4 in this release of the simulator, so the MobiWAN extension, as well as a patch to closely model the Binding Update timings as described in [14] have been ported[2]. DCCP support is not natively present either. Another patch[3] had to be applied to the source tree of the simulator to circumvent this.

The simulation scenarii focused on TCP, UDP and DCCP. A landscape of 800x1600 m has been used, and NS-2 was configured to simulate a regular 11 Mbps 802.11b wireless channel. Some other parameters also had to be adjusted to get the desired simulation conditions. Most notably, the TCP window size had to be set to 64 bytes, which is a much more usual value in current implementations than the default of 20, the wireless reception threshold has been fine tuned to simulate the desired Wi-Fi range (400 m), and all kinds of route optimizations for MIPv6 have been disabled. All the configuration changes are summarized in Table 5.1 on the following page.

Fig. 5.1 on the next page presents the simulated network environment, consisting of one backbone router and three base stations providing non-overlapping wireless connectivity to a mobile node receiving traffic. Part of the simulations were also run with the second base station disabled in order to observe the behavior of the data stream in the case of a more sporadically available network coverage.

[1] http://www.isi.edu/nsnam/ns/

[2] http://mobqos.ee.unsw.edu.au/~omehani/ns/ns-232-mobiwan-103.patch, http://mobqos.ee.unsw.edu.au/~omehani/ns/ns-232-mobiwan-rfc3775.patch

[3] http://lifc.univ-fcomte.fr/~dedu/ns2/dccp-ns2.31.patch

NS-2 parameter	Value
Phy/WirelessPhy bandwidth_	11Mb
Phy/WirelessPhy freq_	2.472e9
Mac/802_11 dataRate_	11Mb
Mac/802_11 basicRate_	1Mb

NS-2 parameter	Value
Phy/WirelessPhy RXThresh_	5.57346e-11
Agent/TCP window__	64
Agent/MIPv6/MN bs_forwarding_	0
Agent/MIPv6/MN rt_opti_	0

Table 5.1: Parameter changes from NS-2 defaults. Top: values to simulate an 802.11b 11 Mbps channel in NS-2. Bottom: miscellaneous changes needed for the simulations.

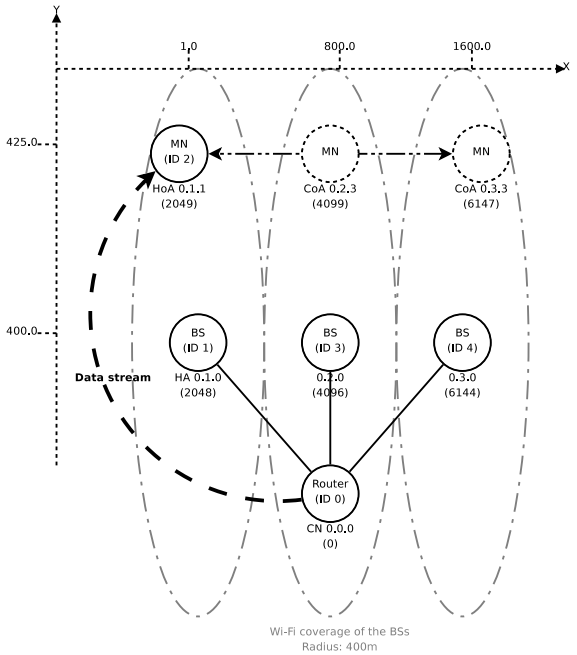

Figure 5.1: The basic simulation scenario: mobile node MN moves back and forth between three adjacent (but not overlapping) wireless networks with different prefixes. Correspondent node CN sends a constant stream of data to MN using the desired transport protocol. Addresses are expressed in NS-2 format.

In the case of DCCP, using CCID3, the *a priori* expected behavior is depicted in Fig. 5.2. A similar behavior was expected from TCP as it also features congestion avoidance algorithms.

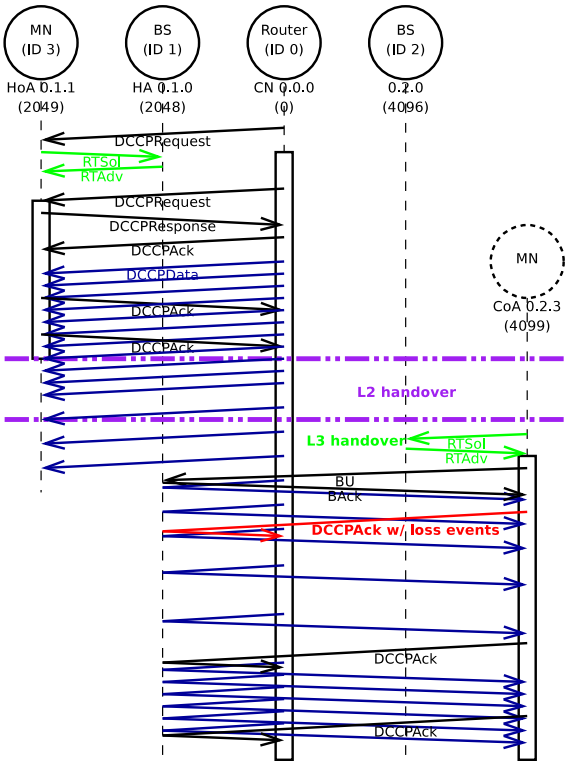

Figure 5.2: The expected packet sequence for a DCCP/TFRC in the vicinity of a handover. The congestion control mechanism mistakenly considers the loss events as a congestion and reduces its sending rate consequently, but inadequately.

Depending on whether the mobile node is on its home link or visiting a foreign link, variables delays ($7 \leq d \leq 15$) are experienced, as shown on Fig. 5.3 on the next page. These are mostly due to the variable length of the path between the mobile host and its correspondent peer, including triangular routing when not at home.

UDP traffic

The data stream is generated using a constant bit rate (CBR) traffic generator. As there is no congestion control mechanism, data is sent by the sender no matter what the receiver's situation is (Fig. 5.4 on the following page shows the results when only two base stations were used, as they are the most informative). This makes UDP the most "reactive" to the re-establishment of the connectivity after the binding update. This is illusory and only due to the fact that the sender perpetually transmits, which means that packets are available to be delivered as soon as the MH can receive data again, or dropped by the intermediate routers otherwise. This also means that bandwidth along the network path is wasted during handovers, as packets only get dropped after having traveled all the way from the sender to the wireless network in which the receiver recently was.

TCP traffic

Short disconnections are experienced due to the handoffs, but are recovered from relatively fast in the case of adjacent networks (Fig. 5.5 on page 21). Figure 5.6 on page 22 is a close-up on the first

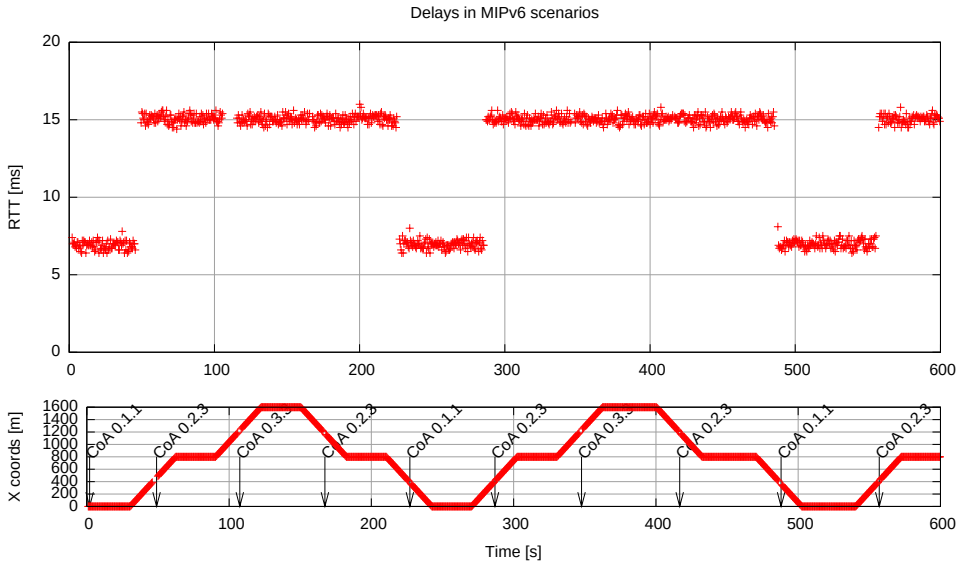

Figure 5.3: RTT measured using the Ping agent from CN to MN. The arrows labeled "CoA" show the time when a Binding Update with a new CoA has been received at the HA, and the tunnel has been updated accordingly.

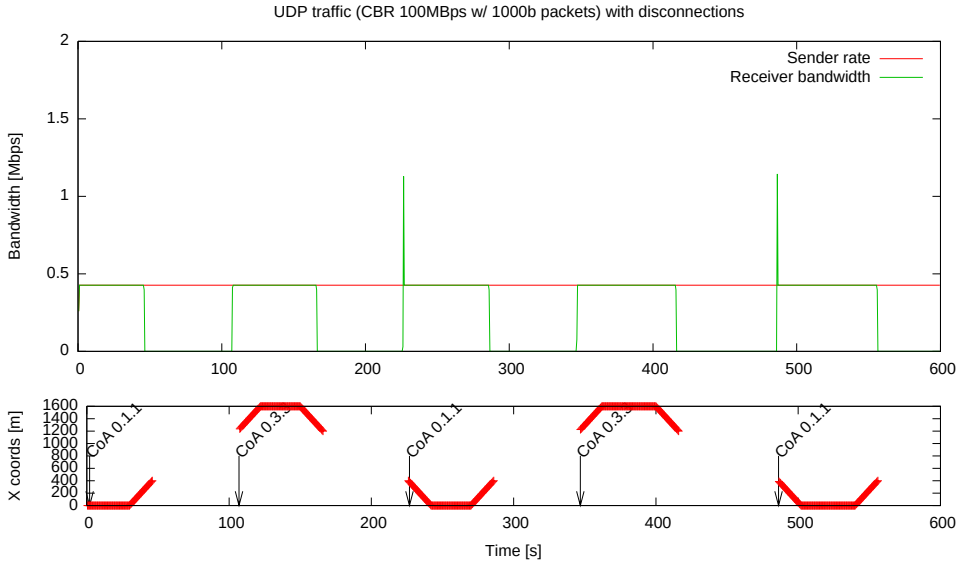

Figure 5.4: An UDP stream between CN and MN when only two base stations are available. Even though MN is not able to receive data all the time, CN does not change its sending rate at all. Incidentally, this is the reason why MN receives data as soon as its binding has been updated.

handover (which is also the longest in this simulation). One can see that the actual handover has been completed in about 3 seconds whereas the TCP stream is only re-established 3 to 4 seconds after a the Binding Update with the new CoA has been received at the Home Agent. This is due to the exponential backoff of TCP, which retries to transmit data periodically with an increasing inter-attempts delay, and fails to immediately detect that network connectivity is available again. This effect is even more noticeable in the case when the second base station is not present. As shown in Fig. 5.7 on the next page, it sometimes takes more than 50 seconds for the data stream to be re-established. It is also important to notice that the data exchange is restarted in slow start mode.

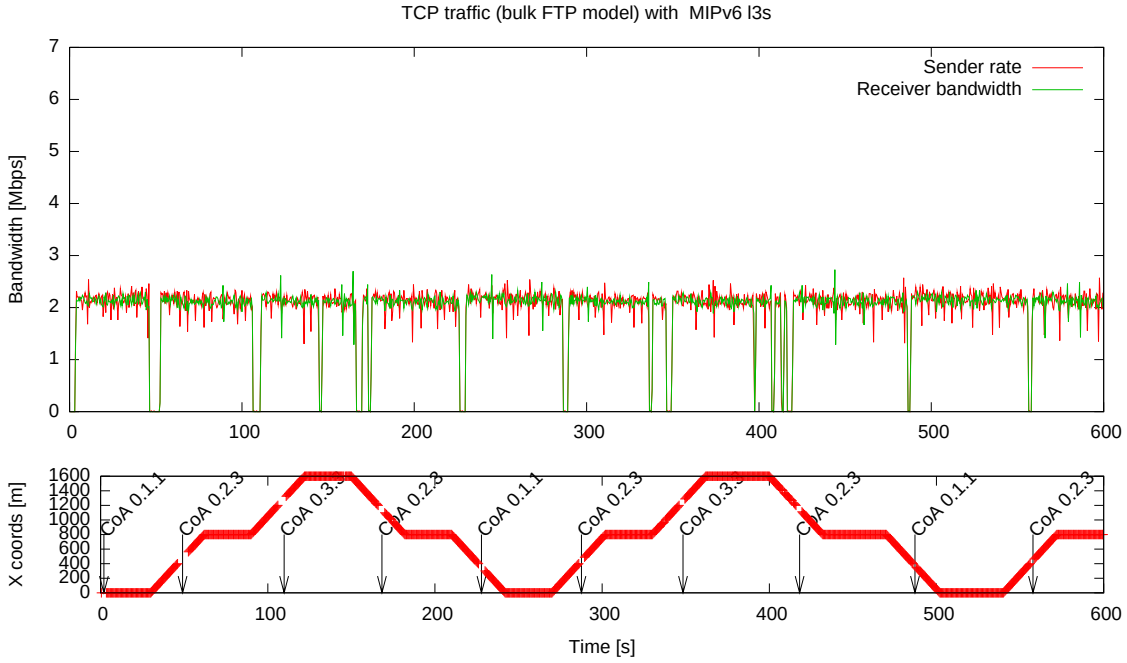

Figure 5.5: A TCP data stream from CN to MN while it is moving between three base stations. The handoff-induced delays are noticeable but do not seem to have a big impact on the connection.

DCCP traffic

As seen on Figure 5.8 on page 23, a similar behavior as TCP is observed for DCCP traffic. This also confirms the expected behavior sketched in Fig. 5.2 on page 19 and is better seen on Fig. 5.9 on page 24.

5.3 Mobility-aware Extension to DCCP: Freeze-DCCP

5.3.1 Brief Overview of Freeze-TCP

To tackle the problem caused (amongst others) by mobility situations to TCP's congestion control mechanisms, some solutions have been proposed. A particularly interesting one is the addition of the possibility for a receiver to "freeze" a sender's stream. This flavor of TCP has been dubbed Freeze-TCP [45] by its authors.

To avoid overloading a receiver's buffers, current implementations of TCP include a flow control mechanism based on a *sliding window*. This window represents the free buffers' size on the receiving side. Without updates, the TCP sender will send at most a number of unacknowledged bytes equal

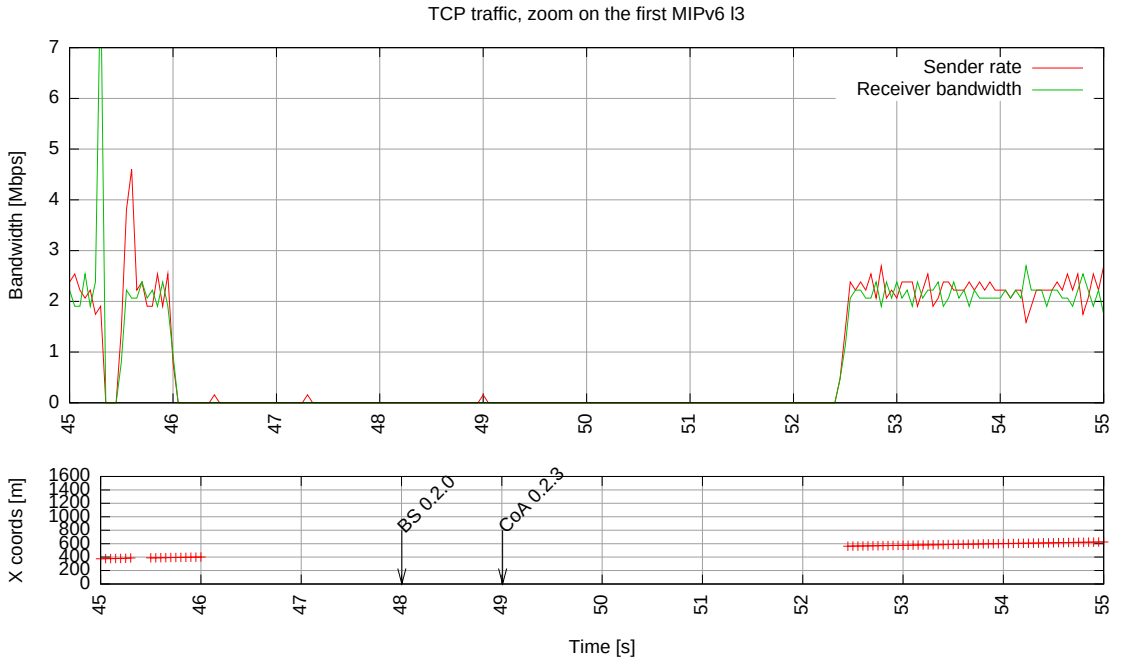

Figure 5.6: Closeup on the first handover of Fig. 5.5 on the previous page. However the layer 2+3 handover is completed in 3 seconds (the arrow labeled "BS" show the end of the layer 2 handoff with the new base station), it takes more than 6 seconds for the TCP stream to restart. Sender attempts to re-initiate the transmission, with an exponential backoff, can be seen at $t \simeq 46.5, 47.3, 49$, when the layer 3 handover is not complete yet, before the traffic can actually restart at $t \simeq 52.4$.

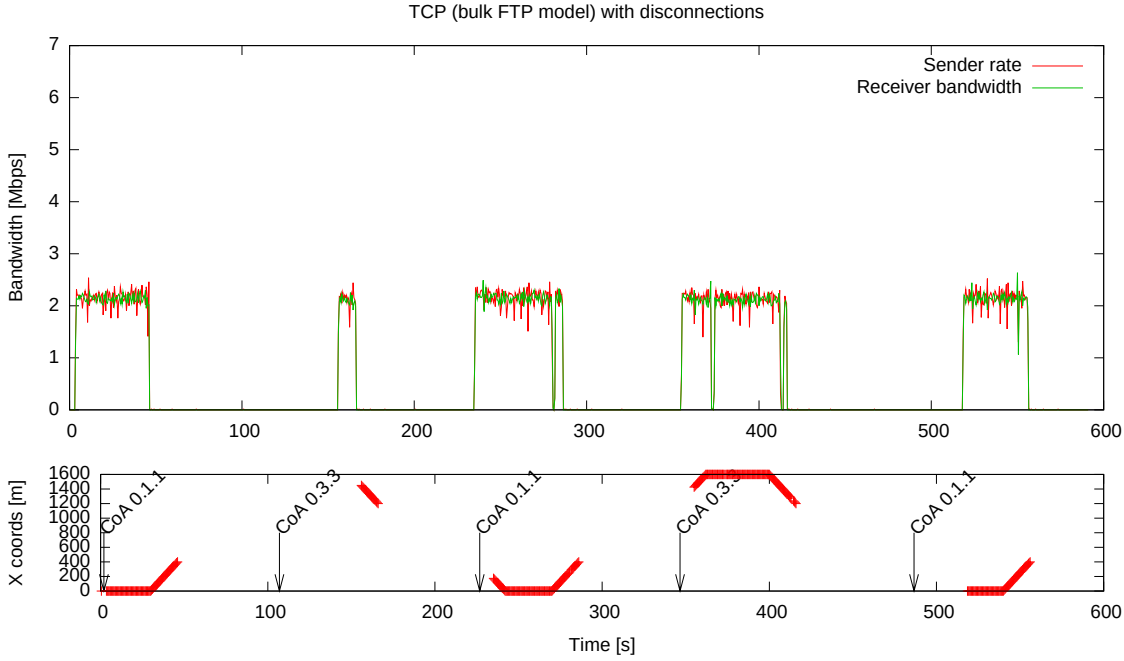

Figure 5.7: A TCP data stream from CN to MN while it is moving between only two base stations. The MIPv6 handovers have now a noticeably bad impact on the traffic.

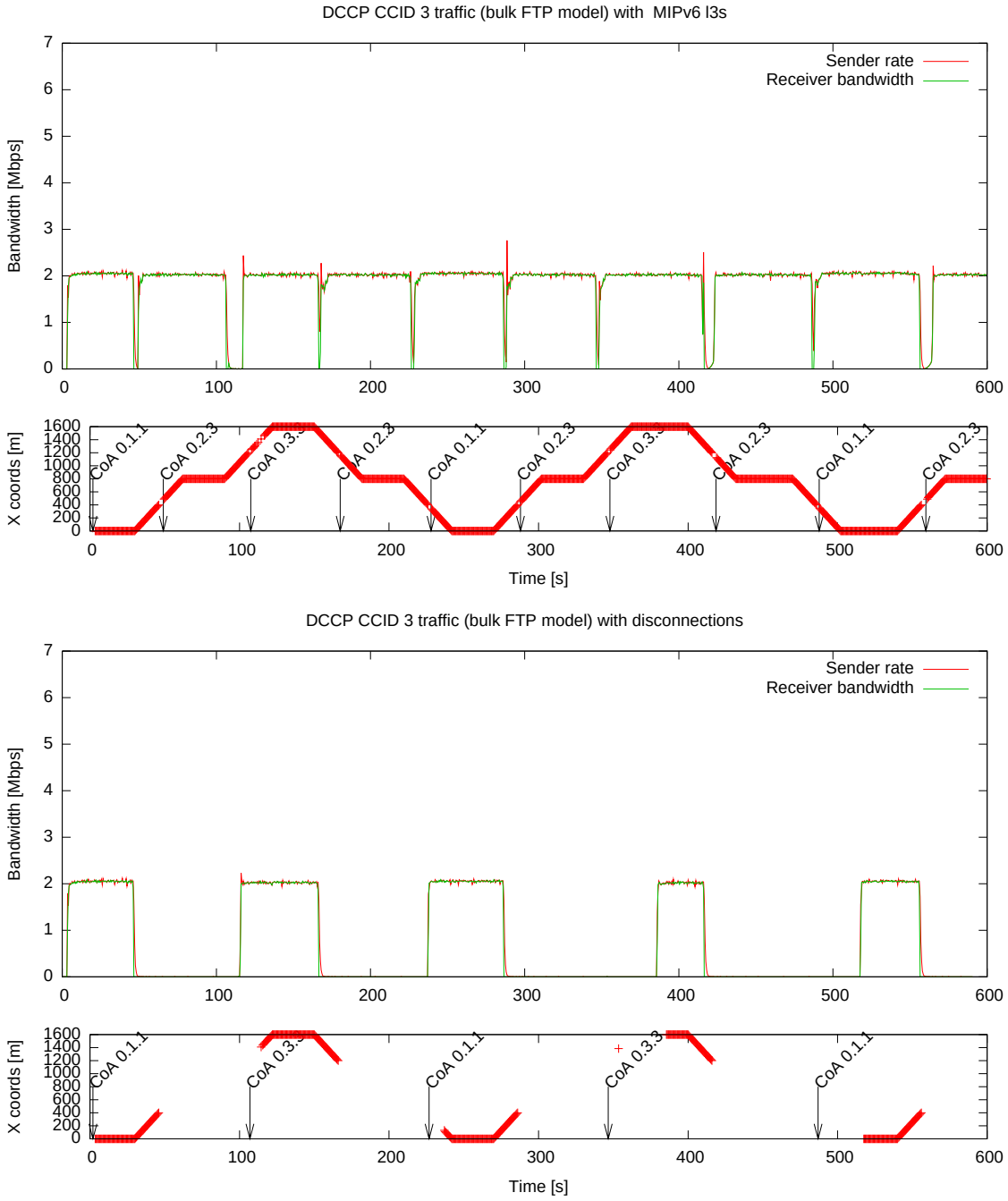

Figure 5.8: Top: DCCP/CCID3 data stream moving through adjacent wireless networks. Bottom: DCCP/CCID3 behavior in a sporadically connected scenario. The behavior of this transport protocol in mobility scenarii is close to that of TCP in the same conditions.

Figure 5.9: A very close zoom of the end of the first DCCP handover with 3 base stations. One can see the initial slow start ($49.08 \leq t \leq 49.17$) due to the wrong assumption that the previous packet losses were due to a congestioned network. The algorithm then overshoots the maximum available bandwidth ($t = 49.19$) and subsequently adjusts its rate ($t \geq 49.4$).

to the current window size. With every acknowledgment, the receiver informs the sender of its buffers' status by updating the window size.

Freeze-TCP proposes a *receiver-side only* modification to force a *Zero Window Advertisement* instead of the size of the current buffer space available. This has the desired consequence, when an acknowledgment carrying this information is received at the sender, to temporarily stop the transmission.

With the appropriate amount of contextual information, a mobile node can then proactively instruct a TCP sender to suspend its data stream, *e.g.* in the situation of a wireless channel fading or before starting a handover. The connection being frozen, no packet losses can happen during the time when the mobile node is unreachable, hence not triggering the sender's congestion avoidance algorithms. When the channel conditions are back to a usable level (or after the successful completion of a handover), the receiver can then send a *Non-zero Window Advertisement* to resume the data stream. An evaluation of this scheme can be found in [52].

Freeze-TCP is proposed in the situation where the receiving node is mobile and subject to predictable disconnections, while the sender is static with an always-on uplink to the rest of the network. The reciprocal scenario does not, however, seem to cause more problems. Considering that a TCP connection can be quiescent for a long time before being closed due to a timeout (Linux 2.6.25 defaults to sending keep-alive packets to maintain a silent connection only 2 hours after the last transmission), it seems reasonable that a TCP sender can be locally frozen (*e.g.* by forcing the window to 0, to keep the same semantics) without informing the receiver when network conditions are such that it is impossible to send data.

Aside from the benefit of avoiding unneeded congestion avoidance behaviors, this extension of the TCP protocol as several other advantages. First, this is an end-to-end extension with no need for support in the infrastructure whatsoever. Additionally, modifications need be made only to the mobile node's stack, which can then transparently interact with all other, already widely deployed, TCP stacks (still benefiting from the improvements).

5.3.2 Applying Freeze-TCP Concepts to DCCP

Based on the aforementioned observation that Freeze-TCP manages to appropriately mitigate the effects of predictable disconnections on the congestion control algorithms, it seems worthwhile to replicate such a behavior for datagrams. DCCP, due to its closest semantics to UDP and the fitness of TFRC for smooth-rate media streaming as well as its adaptation to smaller packets, has been chosen as the datagram-based transport protocol to prioritely work with.

Neither of DCCP's flow control algorithm are receive-window-based[4] as is TCP. Thus, implementing "Freeze-DCCP" is not as straightforward as it was for TCP. Additionally, modifications on both the receiver *and* the sender will be needed. However this second obligation hinders the deployment of this feature over already existing DCCP stacks, it is not seen as a big issue since DCCP is not widely used yet. Moreover, implementation of the following shall make sure that any introduced modification could be ignored by a "classical" DCCP stack without yielding worse performances than with two classical clients.

TFRC relies on several values, either locally computed or transmitted back by the receiver, to calculate its allowed sending rate. Table 5.2 on the next page lists those parameters. In case a packet loss happens, the sender will report values to the sender which would lead it to reduce its sending rate.

Unlike Freeze-TCP, most of the freezing logic would live in the sender part. When instructed to freeze, the proposition is that the DCCP sender saves the "good" values of the rate-controlling variables, and forces its sending rate to 0. On an unfreeze command, the sender would restore the saved parameters and resume the communication at the previous rate. This behavior is shown in the leftmost diagram of Fig. 5.10 on the next page. The rightmost panel shows the options exchanges that should happen when the decision that the connection should be frozen is made on the receiver side.

[4] not to be mistaken with the *congestion control* window, *cwnd* on which CCID2 relies.

Variable	Description	Supplier
RTT_{hist}	history of recent Round Trip Times (used for R and t_{RTO})	measured by the sender
p	loss event rate	reported by the receiver
X_{recv}	receive rate in the last RTT	reported by the receiver
X_{calc}	sending rate from the TCP throughput equation[a]	calculated by the sender
X	allowed sending rate ($\max(X_{\text{calc}}, 2 \cdot X_{\text{recv}})$)	selected by the sender

[a] Eq. 3.1 on page 13

Table 5.2: Variables of the TFRC algorithm influencing the final sending rate.

It is important to note that the proposed behavior makes the strong assumption that both the visited networks (and network paths leading to them) have similar offered capacities and congestion levels. This is a first approach, and networks with different orders of available bandwidth (be it higher or lower) should be addressed using extensions such as Faster Restart [53] or Quick Start [54].

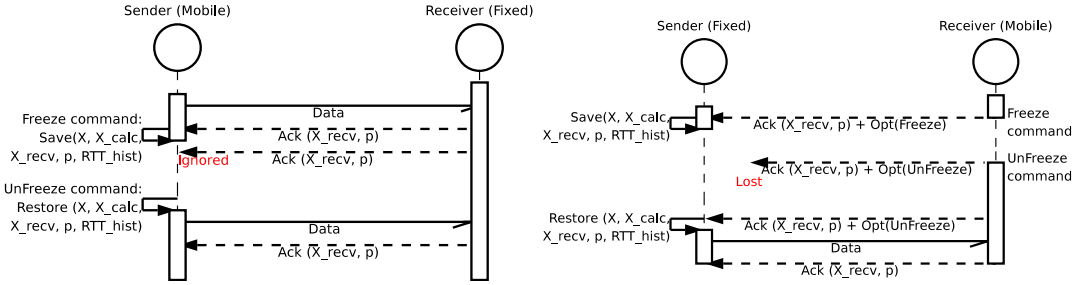

Figure 5.10: Proposed operation for Freeze-DCCP. Left: freeze behavior locally initiated at the sender. Right: options exchange between peers for a mobile receiver to (un)freeze its sender.

When the frozen state has been locally triggered, it is not desirable that a remote option unfreezes the sender. Similarly, a local unfreeze command should not move the sender out of a frozen state requested by the receiver. It is important to differentiate between locally and remotely frozen states. Fig. 5.11 on the following page shows a draft of a state chart representing the additional states of the sender. In certain scenarii, in which both the sender and the receiver are mobile, it is possible that they go through a handover at overlapping times. This would result in the sender being instructed to freeze twice. When facing this situation, the frozen state should be left only when both peers have renewed their connection to the network.

Other issues will certainly be raised, such as a packet sporting an `UnFreeze` option being lost due to the sender's being handing off from one network to the other at the time of delivery.

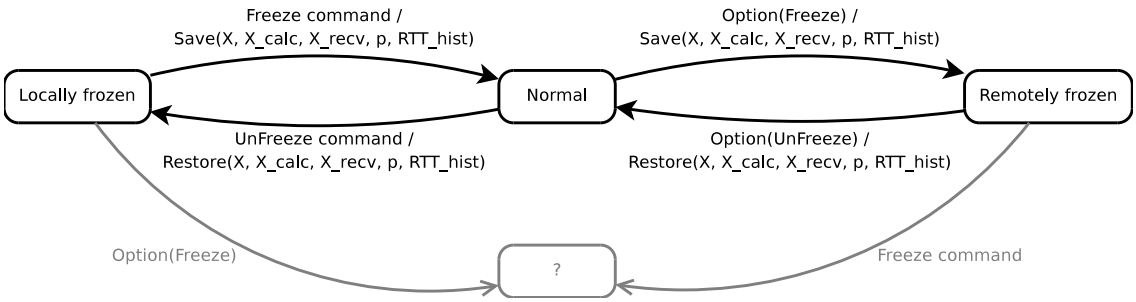

Figure 5.11: State chart of the Freeze-DCCP sender. Two different states are used depending on which side triggered the frozen state. Additionally, another state may be needed to handle more complex situations (*e.g.* both the sender and the receiver are mobile).

Chapter 6

Future Plans

This chapter introduces a tentative programme to progress on the approach proposed in section 4.4. It is expected that the following may more or less span over the next 6 months.

6.1 Implementation and Evaluation of Freeze-DCCP

The DCCP extension proposed in section 5.3.2 has to be evaluated to check that it performs as desired. This will be done in several steps.

Simulations The NS-2 implementation of DCCP will be modified to include Freeze-DCCP agents. These would be used in simulations to evaluate the gained benefit of the proposed scheme in mobility scenarii.

Real implementation Once the algorithm of the simulated agent (in terms of parameters to save, the way to restore them, etc.) would have been validated as functional, an implementation in a real operating system will be written.

Experiments Freeze-DCCP will then be evaluated in real conditions (with mobility). This would most probably be done with VoIP-generated traffic, but non-interactive media streaming would give an additional valuable insight about the overall behavior of the proposed extension.

Generalization As mentioned in section 5.3.2, some strong assumptions have been made about the condition in which Freeze-DCCP maybe operated. Some of the proposed extensions (and maybe others) to remove these assumptions will be integrated.

6.2 Evaluation of the Validity of an External Cross-Layering Entity

While working on the Freeze-TCP NS-2 module, and the idea of Freeze-DCCP, it emerged that an interesting approach to exploit information from other layers would be to rely on a specific entity not standing in the protocol stack, but beside it. This would be, in a way, a generalization of the idea behind the FMIPv6 daemon (Fig. 6.1 on the next page).

One of the most interesting arguments for taking this entity out of the network stack is that it could then be used to set some generic behavioral policies for network protocols without without unnecessarily adding to the complexity of specific protocols. It could also have access to relevant but non network-related information which can be of use in determining the communication profile to adopt.

Work on how such an entity should interact with the protocols stack, how it should be implemented, and how well it eventually performs shall be done in depth.

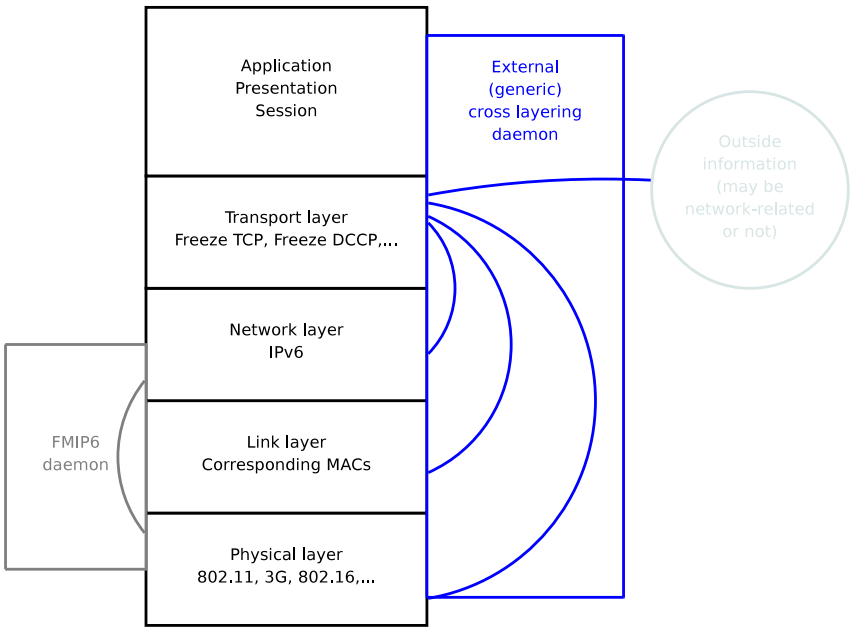

Figure 6.1: An external cross-layering entity would be able to appropriately provide every layer with valuable information about the state of the other *and* even non-network related states (geographic position, current context, ...) to improve the performances of the protocol stack altogether.

6.3 Deeper Considerations About Service Discovery

A more comprehensive analysis and evaluation of the available service discovery protocols for pervasive networking shall be performed. An implementation of that (or those) which turns out to be the most appropriate will be used in the embryo of a pervasive networking testbed in which various scenarii are supposed to be experimented and evaluated afterwards.

Bibliography

[1] Fen Zhu, Matt W. Mutka, and Lionel M. Ni. Service discovery in pervasive computing environments. *IEEE Pervasive Computing*, 4(4):81–90, 2005.

[2] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service location protocol, version 2. RFC 2608 (Proposed Standard), June 1999. Updated by RFC 3224. Online: http://tools.ietf.org/html/rfc2608.

[3] E. Guttman. Service location protocol modifications for IPv6. RFC 3111 (Proposed Standard), May 2001. Online: http://tools.ietf.org/html/rfc3111.

[4] S. Cheshire and M. Krochmal. DNS-based service discovery, 2006. Online: http://tools.ietf.org/html/draft-cheshire-dnsext-dns-sd-04.

[5] S. Cheshire and M. Krochmal. Multicast DNS, 2006. Online: http://tools.ietf.org/html/draft-cheshire-dnsext-multicastdns-06.

[6] Y. Y. Goland, T. Cai, P. Leach, and Y. Gu. Simple service discovery protocol/1.0, 1999. Online: http://tools.ietf.org/html/draft-cai-ssdp-v1-03.

[7] M. Abou El Saoud, T. Kunz, and S. Mahmoud. SLPManet: service location protocol for MANET. In *IWCMC '06: Proceedings of the 2006 Int'l Conference on Wireless Communications and Mobile Computing*, pages 701–706, New York, NY, USA, 2006. ACM.

[8] M. Krebs, K.-H. Krempels, and M. Kucay. Service discovery in wireless mesh networks. In *WCNC 2008: Wireless Communications and Networking Conference, 2008*, 2008.

[9] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). RFC 3626 (Experimental), 2003. Online: http://tools.ietf.org/html/rfc3626.

[10] A. Laouti, P. Jacquet, P. Minet, L. Viennot, T. Clausen, and C. Adjih. Multicast optimized link state routing. Technical report, Inria, 2003.

[11] F. Nazir and A. Seneviratne. Towards mobility enabled protocol stack for future wireless network. *Ubiquitous Computing and Communication Journal*, 2(4), August 2007.

[12] W. M. Eddy. At what layer does mobility belong? *IEEE Communications Magazine*, October 2004.

[13] C. Perkins. Ip mobility support for IPv4. RFC 3344 (Proposed Standard), August 2002. Updated by RFC 4721. Online: http://tools.ietf.org/html/rfc3344.

[14] D. Johnson, C. Perkins, and J. Arkko. Mobility support in IPv6. RFC 3775 (Proposed Standard), 2004. Online: http://tools.ietf.org/html/rfc3775.

[15] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert. Network mobility (NEMO) basic support protocol. RFC 3963 (Proposed Standard), January 2005. Online: http://tools.ietf.org/html/rfc3963.

[16] R. Jain, J. Burns, M. Bereschinsky, and C. Graff. Mobile IP with location registers (MIP-LR), 2001. Online: http://tools.ietf.org/html/draft-jain-miplr-01.

[17] M. Kunishi, M. Ishiyama, K. Uehara, and H. Teraoka. LIN6: A new approach to mobility support in IPv6. In *Int'l Symposium on Wireless Personal Multimedia Communication*, 2000.

[18] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host identity protocol. RFC 5201 (Experimental), April 2008. Online: http://tools.ietf.org/html/rfc5201.

[19] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: Congestion control without reliability. In *SIGCOMM '06*, September 2006.

[20] E. Kohler, M. Handley, and S. Floyd. Datagram congestion control protocol (DCCP). RFC 4340 (Proposed Standard), March 2006. Online: http://tools.ietf.org/html/rfc4340.

[21] E. Kohler. Generalized connections in the datagram congestion control protocol, 2006. Online: http://tools.ietf.org/html/draft-kohler-dccp-mobility-02.

[22] D. A. Maltz and P. Bhagwat. MSOCKS: An architecture for transport layer mobility. In *INFOCOM (3)*, pages 1037–1045, 1998.

[23] A. C. Snoeren and H. Balakrishnan. TCP connection migration, November 2000. Online: http://tools.ietf.org/html/draft-snoeren-tcp-migrate-00.

[24] D. Funato, K. Yasuda, and H. Tokuda. TCP-R: TCP mobility support for continuous operation. In *ICNP '97: Proceedings of the 1997 International Conference on Network Protocols*, page 229, Washington, DC, USA, 1997. IEEE Computer Society.

[25] W. Eddy. Mobility support for TCP, April 2004. Online: http://tools.ietf.org/html/draft-eddy-tcp-mobility-00.

[26] A. Bakre and B. R. Badrinath. I-TCP: indirect TCP for mobile hosts. In *International Conference on Distributed Computing Systems*, page 136, Vancouver, Can, 1995. IEEE, Piscataway, NJ, USA.

[27] R. Stewart. Stream control transmission protocol. RFC 4960 (Proposed Standard), September 2007. Online: http://tools.ietf.org/html/rfc4960.

[28] B. Landfeldt, T. Larsson, Y. Ismailov, and A. Seneviratne. SLM, a framework for session layer mobility management. In *ICCCN 99*, 1999.

[29] J. Tourrilhes. L7-mobility : A framework for handling mobility at the application level. In *PIMRC 2004*, 2004.

[30] H. Schulzrinne and E. Wedlund. Application-layer mobility using SIP. *SIGMOBILE Mobile Computing and Communications Review*, 4(3):47–57, 2000.

[31] R. Wakikawa, V. Devarapalli, T. Ernst, and K. Nagami. Multiple care-of addresses registration, May 2008. Online: http://tools.ietf.org/html/draft-ietf-monami6-multiplecoa-08.

[32] A. Matsumoto, M. Kozuka, K. Fujikawa, and Y. Okabe. TCP multi-home options, October 2003. Online: http://tools.ietf.org/html/draft-arifumi-tcp-mh-00.

[33] 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), 2002. Updated by RFCs 3265, 3853, 4320, 4916. Online: http://tools.ietf.org/html/rfc3261.

[34] R. Sparks. The session initiation protocol (SIP) Refer method. RFC 3515 (Proposed Standard), April 2003. Online: http://tools.ietf.org/html/rfc3515.

[35] N. Montavont and T. Noël. Stronger interaction between link layer and network layer for an optimized mobility management in heterogeneous IPv6 networks. *Pervasive and Mobile Computing Journal*, 2(3):233–261, september 2006.

[36] R. Koodli. Fast handovers for mobile IPv6. RFC 4068 (Experimental), 2005. Online: http://tools.ietf.org/html/rfc4068.

[37] P. McCann. Mobile IPv6 fast handovers for 802.11 networks. RFC 4260 (Informational), November 2005. Online: http://tools.ietf.org/html/rfc4260.

[38] F. Teraoka, K. Gogo, K. Mitsuya, R. Shibui, and K. Mitani. Unified Layer 2 (L2) Abstractions for Layer 3 (L3)-Driven Fast Handover. RFC 5184 (Experimental), May 2008. Online: http://tools.ietf.org/html/rfc5184.

[39] K. Gogo, R. Shibui, and F. Teraoka. An l3-driven fast handover mechanism in ipv6 mobility. In *SAINT-W '06: Int'l Symposium on Applications on Internet Workshops*, pages 10–13, Washington, DC, USA, 2006. IEEE Computer Society.

[40] A. T. Campbell and J. Gomez-Castellanos. IP micro-mobility protocols. *SIGMOBILE Mobile Computing and Communications Review*, 4(4):45–53, 2000.

[41] P. Reinbold and O. Bonaventure. A survey of IP micro-mobility protocols, 2002.

[42] H. Soliman, C. Castelluccia, K. El Malki, and L. Bellier. Hierarchical mobile IPv6 mobility management (HMIPv6). RFC 4140 (Experimental), August 2005. Online: http://tools.ietf.org/html/rfc4140.

[43] A. E. Yegin, M. Parthasarathy, and C. Williams. Mobile IPv6 neighborhood routing for fast handoff.

[44] K. Fall and S. Floyd. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *Computer Communication Review*, 26(3):5–21, July 1996.

[45] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta. Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments. In *INFOCOM (3)*, pages 1537–1545, 2000.

[46] S. Floyd and E. Kohler. Profile for datagram congestion control protocol (DCCP) congestion control ID 2: TCP-like congestion control. RFC 4341 (Proposed Standard), March 2006. Online: http://tools.ietf.org/html/rfc4341.

[47] S. Floyd, E. Kohler, and J. Padhye. Profile for datagram congestion control protocol (DCCP) congestion control ID 3: Tcp-friendly rate control (TFRC). RFC 4342 (Proposed Standard), March 2006. Online: http://tools.ietf.org/html/rfc4342.

[48] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM 2000*, pages 43–56, Stockholm, Sweden, August 2000.

[49] S. Floyd and E. Kohler. Profile for datagram congestion control protocol (DCCP) congestion id 4: TCP-friendly rate control for small packets (TFRC-SP), 2008. Online: http://tools.ietf.org/html/draft-ietf-dccp-ccid4-02.

[50] O. Mehani, R. Benenson, S. Lemaignan, and T. Ernst. Networking needs and solutions at Imara. In Ulrich Finger, Masayuki Fujise, Christian Bonnet, Massimiliano Lenardi, Shozo Komaki, and GuangJun Wen, editors, *ITST 2007, 7th Int'l Conference on Intelligent Transport Systems Telecommunications, Sophia Antipolis, France, June 6-8, 2007*, pages 362–367, June 2007.

[51] M. Tsukada, O. Mehani, and T. Ernst. Simultaneous usage of NEMO and MANET for vehicular communication. In *WEEDEV 2008: 1st Workshop on Experimental Evaluation and Deployment Experiences on Vehicular Networks in conjonction with TRIDENTCOM 2008, Innsbruck, Austria, March 18, 2008*, March 2008.

[52] Adeel Baig, Lavy Libman, and Mahbub Hassan. Performance enhancement of on-board communication networks using outage prediction. *IEEE Journal on Selected Areas in Communications*, 24(9):1692–1701, 2006.

[53] E. Kohler, S. Floyd, and A. Sathiaseelan. Faster restart for TCP friendly rate control (TFRC), 2007. Online: http://tools.ietf.org/html/draft-ietf-dccp-tfrc-faster-restart-05.

[54] G. Fairhurst and A. Sathiaseelan. Quick-start for datagram congestion control protocol (DCCP), 2008. Online: http://tools.ietf.org/html/draft-fairhurst-tsvwg-dccp-qs-03.