INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2007; **17**: 307–319
Published online 8 February 2007 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/nem.647

IP micromobility and QoS virtual network testbed

Nima Nafisi*,†,[1], Lin Wang[1], Hamid Aghvami[1], Ramon Ferrús[2] and Xavier Revés[2]

[1]*Centre for Telecommunications Research, King's College, London, UK*
[2]*Signal Theory and Communications Department UPC, Barcelona, Spain*

SUMMARY

This paper examines issues related to the interaction of QoS and IP micromobility management. More precisely, the inter-working of a tunnel-based IP micromobility management with the bandwidth broker QoS control plane is considered. Furthermore, the implementation of the system is detailed and it is tested using a virtual network testbed, based on 'user-mode-Linux'. Finally, as a proof of concept the signalling between the different entities is examined with the protocol analyser 'ethereal'. Copyright © 2007 John Wiley & Sons, Ltd.

1. INTRODUCTION TO IP MICROMOBILITY

In order to improve the performance of the IP mobility management provided by Mobile IP [1], different micromobility protocols have been proposed [2]. With Mobile IP each time a *mobile node* (MN) changes its IP attachment point, signalling has to be exchanged between the MN and its *correspondent node* (CN) in order to establish an IP-in-IP tunnel. This incurs delay, packet loss and signalling overhead. Micromobility protocols have been introduced in order to manage locally the mobility of users inside an access network. Thus, Mobile IP functions are confined to macromobility, i.e. the mobility between domains, and are transparent to the micromobility used inside a domain. Micromobility protocols can be classified into two groups:

- Tunnel-based protocols, for which incoming packets are read at the *anchor point* (ANP) (Figure 1); a lookup for the MN address is then done in the visitor list, and finally the packet is tunnelled toward the appropriate router. The end of the tunnel, which is the *access router* (AR) to which the MN is attached, is usually one or more hops away from the entry situated at the gateway. There are basically two protocols based on this mechanism: HMIP [3] and BCMP [4,5].
- Host-based forwarding protocols, for which incoming packets are forwarded based on a location database that maps the host identifier to location information, in a similar way as the precedent class, but here the incoming packets at the gateway are forwarded to a router which is one hop away. This process of forwarding continues until the packet reaches the mobile host. Therefore, in this protocol class each router in the access network has an MN location database. We can mention two examples for this class: Cellular IP and HAWAII [6].

After having reviewed the micromobility classes, we will show how a tunnel-based IP micromobility management (BCMP) and QoS entities (bandwidth broker, DiffServ edge routers) interact. This interaction occurs when the user performs a handover to a new IP attachment point, as the availability of DiffServ resources along the new path has to be checked by the bandwidth broker and edge routers belonging

*Correspondence to: N Nafisi, Centre for Telecommunications Research, King's College, London WC2R 2LS, UK.
†E-mail: nima.nafisi@kcl.ac.uk
Contract/grant sponsor: Project IST-AROMA, European Community

Received 31 October 2006
Accepted 10 December 2006

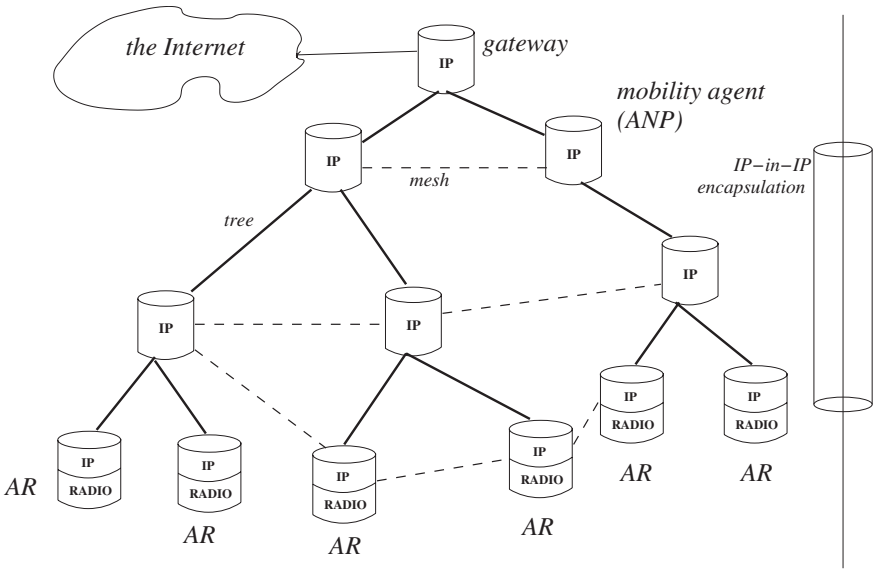

Figure 1. IP mobile access network

to the path have to be reconfigured. Thus, first the BB functions and then the description of the testbed implementation are presented.

2. BANDWIDTH BROKER

2.1 SLA/SLS

The *service level agreement* (SLA) is a general term for designating the agreement between a service provider and a customer. It includes terms and conditions of the agreement and also a set of technical parameters, called *service level specification* (SLS). The specificity of DiffServ is that it does not guarantee QoS at the flow level but only at the level of aggregated flows. Flows with the same DiffServ code-point form an aggregation of flows, which experience statistically the same QoS level defined by their SLS. The forwarding treatment provided at a router to aggregated flows is referred to as *per hop behaviour* (PHB). PHBs are not strictly defined; they should only be the same within one DiffServ domain.

There are two PHBs, which are defined:

- expedited forwarding PHB for real-time traffic;
- assured forwarding PHB for elastic traffic.

The different levels of service specification for the DiffServ architecture are shown in Figure 2. An IETF draft [7] has been issued, which depicts the content of an SLS for the DiffServ framework. The attributes of the SLS template as presented in the draft are as follows:

- The *scope attribute*: indicates where the QoS policy is enforced. In the SLS, a couple of ingress and egress routers should be given.

  Scope = (ingress, egress), where ingress (and egress) is an interface id, a set of interface ids, or an unspecified interface.

- The *flow id attribute* = {DiffServ information, source information, destination information, application information}, where the application information can be the port (source or destination) or unspecified.

DOI: 10.1002/nem

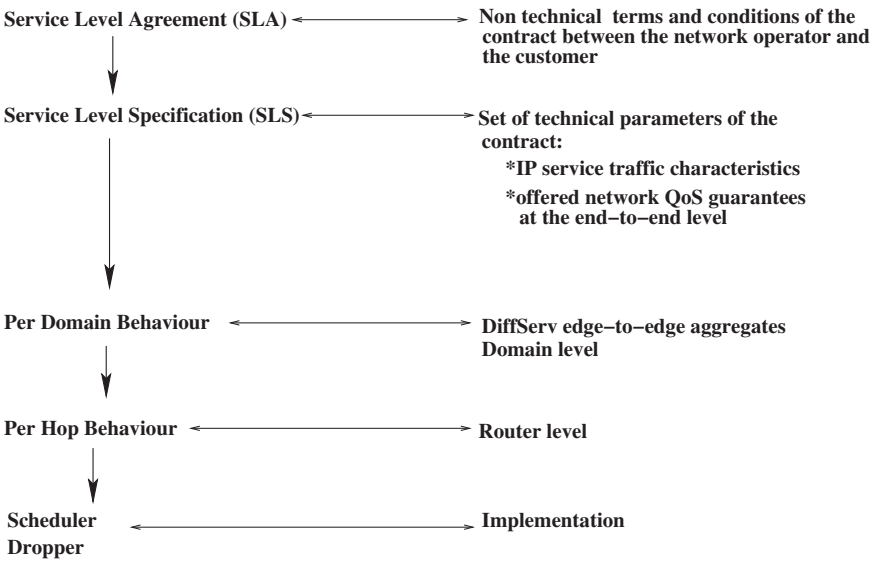

Figure 2. Service specification at different levels for the DiffServ framework

- The *performance attribute*: the network level of quality associated to the IP flow. Four indicators are used:
  —one-way delay inter-packet;
  —delay variation;
  —packet loss rate;
  —throughput.
- If no quantitative indicators have been specified, qualitative indicators can be used like 'low, medium, high', which correspond to the QoS IP-based service offering such as 'Bronze, Silver, Gold'.
- The *traffic conformance attribute*: a set of indicators that describe the ingress IP flow traffic. This set is composed for instance by:
  —peak rate;
  —bucket depth;
  —maximum transmit unit;
  —minimum packet size.

With this set, the conformance test would be based on the token bucket algorithm. After the test, 'in-profile' packets are tagged with the negotiated DiffServ code-point and 'out-profile' packets are either dropped, shaped, or marked with a lower-priority DiffServ code-point.

2.2 Internet2 architecture of the BB

The concept of BB appeared in Neilson *et al.* [8], and its functionalities have been further specified by the Internet2 Qbone BB working group. The BB concept is a centralized solution to the QoS management problem of a DiffServ domain (nevertheless, more distributed approaches with a hierarchy of BBs have also been proposed [9]). The BB controls the network resources in its domain, by giving different priorities to aggregated flows following the DiffServ principle, and also interacts with QoS managers in neighbour domains for QoS negotiation and reservation. The following main BB functionalities have been specified by the Internet2 Qbone WG:

- *Inter-domain communication* interface with adjacent cooperating BBs. A protocol named SIBBS (Simple Interdomain BB Signalling Protocol) has been specified by the WG.

DOI: 10.1002/nem

- *Intra-domain communication* interface: a communication method is needed for the BB to configure and allocate resources to the edge routers of the DiffServ domain. Only the edge routers keep QoS related states; the core routers are stateless. The BB has to communicate with the edge routers in order specify how edge routers should classify, condition and mark the entering flows into appropriate aggregated flow classes. For this communication, the following options are available:
  —Telnet: a command line interface that can be used to configure remote routers;
  —SNMP (*Simple Network Management Protocol*): an application layer protocol that allows the exchange of management info between agents residing on a managed device and a network management system. The SNMP SET message could be used for the configuration of edge routers;
  —COPS (*Common Open Policy Service*): the BB acts as a PDP (*policy decision point*) and enforces the DiffServ configuration at the edge router, which is a PEP (*policy enforcement point*).
- Knowledge of the domain and its resources: this function can be achieved by the use of SNMP, which allows monitoring of a set of routers. Another solution is to have an interface to the link-state routing in the domain so that the BB can have knowledge of the network topology. Furthermore, information from the exterior routing is also needed in order to detect the edge routers and to identify adjacent BBs.
- Admission control: a mechanism is required that can evaluate whether requested resources (by a user or an adjacent domain QoS manager) is available in the domain, or more precisely in a set of routers under the control of the BB. Admission control can be done based on the arrival time of each request, i.e., new flows are accepted sequentially until the capacity of the domain is reached; or another method is to practise a policy-based admission control . In a policy-based admission control, the admission is not based only on the availability of the network resources but also on the identity of the user or application, and how, when and where the flow enters the network.

3. VIRTUAL NETWORK TESTBED

The *user-mode-Linux* (UML) kernel [10] can be booted up within a Linux machine as a user-space program; moreover, a complete Linux environment (kernel and user-space applications) can run inside a so-called 'host machine'. Usually, the Linux kernel communicates directly with the hardware on the computer it is running. UML communicates with the host machine kernel, making it possible to run several instances of virtual machines. Multiple instances of UML can be connected together to create a virtual network. A virtual network can be defined as a TCP/IP communication network built upon virtual machines hosted on a single machine. Thus, for layer 3 and above, there is no difference in terms of the communication stacks between a virtual and a 'real' network. While network simulators are used to model the performance of communication networks, virtual networks can be used as a proof of concept and emulate the functional aspects.

UML has different ways to communicate with the host machine or other virtual machines. A possibility is to use the utility called 'uml_switch', which provides communication between virtual machines by means of UNIX domain sockets. Furthermore, in the case of an IP mobile access network, where the MN needs connectivity to ARs with different subnets, the latter utility can be used. Hence, the MN's and ARs' network interfaces being connected to the same uml_switch, the MN would receive the router advertisements of different ARs. The virtual network configuration used for the tests is depicted in Figure 3.

4. BB CONTROL PLANE AND IP MICROMOBILITY INTEGRATION

The BCMP mobility management protocol implementation [11] is a C program, implementing the MN, the anchor point (ANP) and the AR functionalities. The program uses POSIX threads for client/server signalling between MN and AR and between AR and ANP. An IP-in-IP tunnelling for the data packets from the ANP to an AR is implemented in user-space. Finally, the IP handover is triggered manually by

DOI: 10.1002/nem

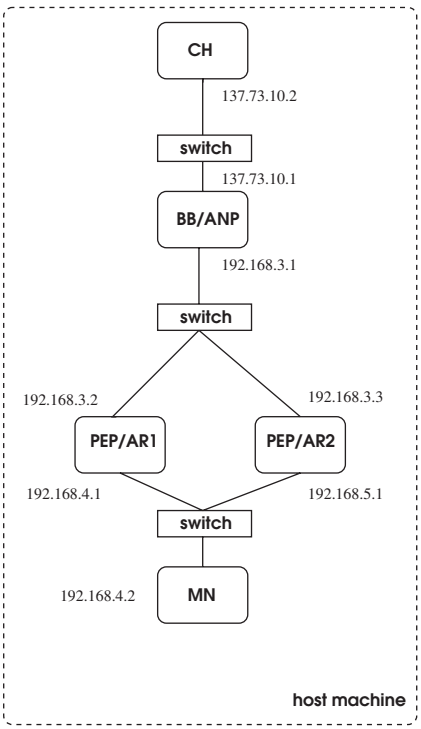

Figure 3. UML virtual network

a command line at the MN, instead of a trigger mechanism based on a set of ARs' signal quality values. For the interaction between the QoS and the mobility management programs, the approach chosen is the use of an AF_UNIX domain socket, which enables inter-process communication with the socket API. However, as the BB program is written in Java [12] and the mobility management program in C, Java Native Interface (JNI) is used in order to enable the Java program (BB) to create an AF_UNIX socket. Hence, for instance before a handover execution, the BCMP program running on the AR node can communicate with the PEP program on the same node in order to trigger a QoS reservation request to the BB. Hereafter, we examine the inter-working between the BCMP micromobility and the BB modules occur. The BCMP protocol can be separated into several functions: initial login, handover preparation and handover execution, detailed in the following subsections.

4.1 'Netfilter'

Before going into more detail regarding IP mobility and QoS implementation, a short insight into the Linux Netfilter mechanism is given, since it is used for the handover preparation function and gives a good overview of the Linux network stack. Netfilter is a packet filtering subsystem in the Linux kernel stack. It consists of five hook functions. These hook functions allow one to analyse packets in various locations in the network stack. The hook functions for the IPv4 and IPv6 packets are, respectively, declared in the kernel source files: *Linux/Netfilter_ipv4.h* and *Linux/Netfilter_ipv6.h*. Figure 4 shows the location of the hook function in the Linux network stack. As shown in Figure 4, the NF_IP_PRE_ROUTING is the first hook called after the packet has been received at the network interface. NF_IP_LOCAL_IN is used for packets destined for the network stack. The NF_IP_FORWARD hook is for packets not destined for the machine but that should be forwarded towards another destination. NF_IP_POST_ROUTING is for packets that have been routed and are ready to be transmitted. Finally, the NF_IP_LOCAL_OUT is for packets generated by the machine itself. Thus, thanks to one of these hook functions, a packet, during

DOI: 10.1002/nem

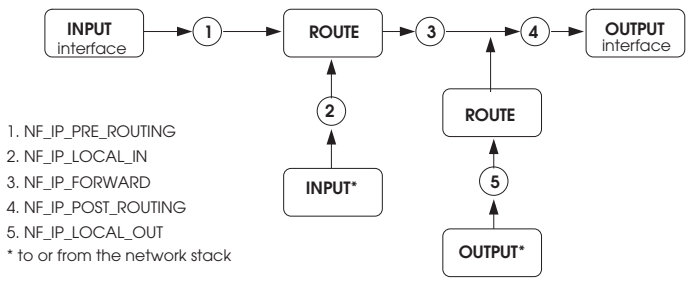

Figure 4. Netfilter hook functions

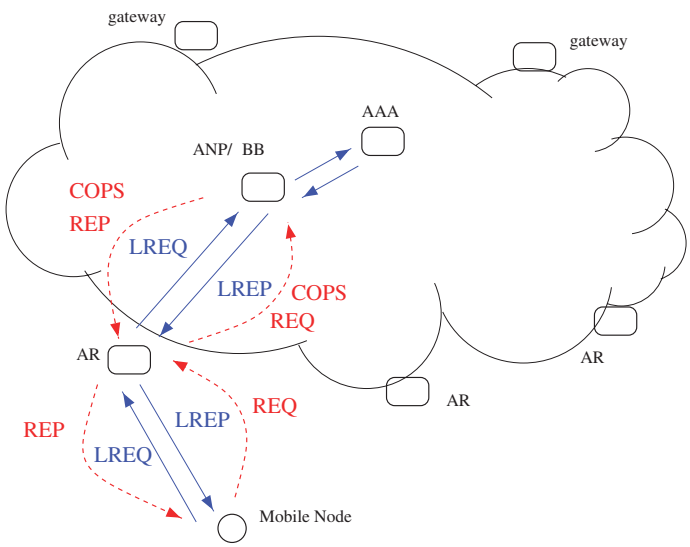

Figure 5. Login procedure

its journey through the network stack, can be filtered and modified. After the modification, the packet can be reinjected into the network stack. The following options for packet reinjection are possible:

- NF_ACCEPT: the packet continues its journey in the stack from the point it has been filtered.
- NF_DROP: the packet is dropped.
- NF_REPEAT: repeat the hook function.
- NF_STOLEN: the packet stops its journey and is not reinjected into the stack.
- NF_QUEUE: the packet is queued towards a user-space program.

4.2 Login

In the login procedure, the MN is authenticated and obtains an IP address (care-of address) from the ANP. In Figure 5, shown with dashed lines is the QoS extension to the BCMP signalling. In addition to the initial login signalling of the micromobility protocol, QoS resources have to be reserved when the MN logs into the network. This reservation signalling can be done sequentially after the login signalling of the micromobility protocol. The MN sends an LREQ (*login request*) message, which contains its home address. This message is then forwarded by the AR to the ANP/BB, which performs the authentication and authorization with the AAA entity. Then, a care-of address is given by the ANP to the MN thanks to the LREQ message. Once the MN has received a care-of address, it requests resources by using a simple

DOI: 10.1002/nem

request/reply signalling. The MN sends a request containing the requested SLS to the current AR to which it is attached. The AR forwards the message to the ANP/BB, which performs admission control for the flow. Then, the BB enforces the QoS decision (DiffServ configuration) into the AR/PEP (*access router/policy enforcement point*). Finally, the AR replies to the MN in order to confirm the set-up.

The initial BCMP mobility implementation uses RAW sockets for IP-in-IP tunnelling between the ANP and the AR. However, when DiffServ is used in conjunction with micromobility, the encapsulated data packets are not marked properly at the ANP if RAW socket IP-in-IP tunnelling is used. As shown in Figure 6, if a RAW socket is used for implementation of the IP-in-IP encapsulation, then the packet is already encapsulated and the DiffServ filtering cannot occur based on the CN and MN IP addresses. If the kernel module 'ipip' is used, the IP-in-IP encapsulation occurs after the routing decision has been taken, i.e. just before the Netfilter hook NF_IP_FORWARD. Therefore, the incoming data packets can be filtered and marked before encapsulation by the ipip kernel module. It can be noticed that the ipip module reproduces in the outer IP header the DSCP found in the inner IP header of the IP-in-IP header.

The issue raised above concerned the DiffServ marking of data packets. Also, the signalling packets (for QoS or mobility management) have to be marked with the highest DSCP in order not to be dropped during congestion time. For this, the signalling packets are marked when they are generated using the 'setsockopt' function. This marking is effective as the signalling packets are injected into the kernel at the Netfilter hook 5 (Figure 4) and the DiffServ forwarding occurs at the output interface.

After the login messages have been exchanged with the ANP and the MN has obtained a care-of address, prior to the establishment of a session a QoS signalling has to be carried out in order to reserve the required resources in the access network. In our testbed, this QoS signalling is based on the COPS protocol. Two different approaches are possible for this QoS signalling between the MN and the BB. Either the BB IP address is known to the MN (through router advertisement) or the MN cannot have access to it, for security reasons, for example. First, if the MN has access to the BB IP address, which should be a routable address, the BB establishes a connection to the BB. However, it may not be possible for the BB to have a routable IP address. In this case, the other possibility is to have a proxy server at the AR which forwards the requests and replies to and from the BB. Since the COPS protocol is used for the signalling, the connection established with the AR is based on TCP. This may be a problem because of issues encountered with TCP in a wireless environment and the need to re-establish a connection after a handover. This connection to the BB via an AR is necessary at the session establishment, and is optional during an IP handover if only the BB modifies the MN's QoS after a handover. Either the TCP connection to the BB is re-established after a handover when needed, or the QoS signalling has to based on a connectionless transport protocol like UDP.

4.3 Handover execution

The handover procedure (Figure 7) is used in order to update the location of the MN. After a successful layer 2 handover, the MN sends a HOFF message to the new AR. In the HOFF message sent by the MN the SLS should be included, so that the AR can check if available resources are sufficient for the handover

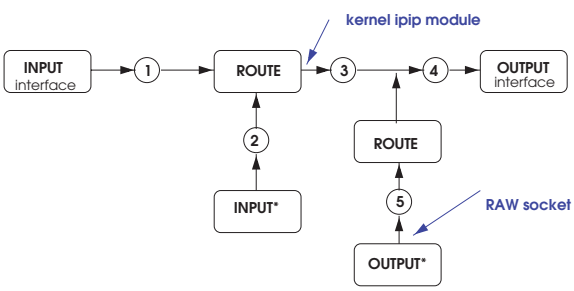

Figure 6. DiffServ and IP-in-IP encapsulation

DOI: 10.1002/nem

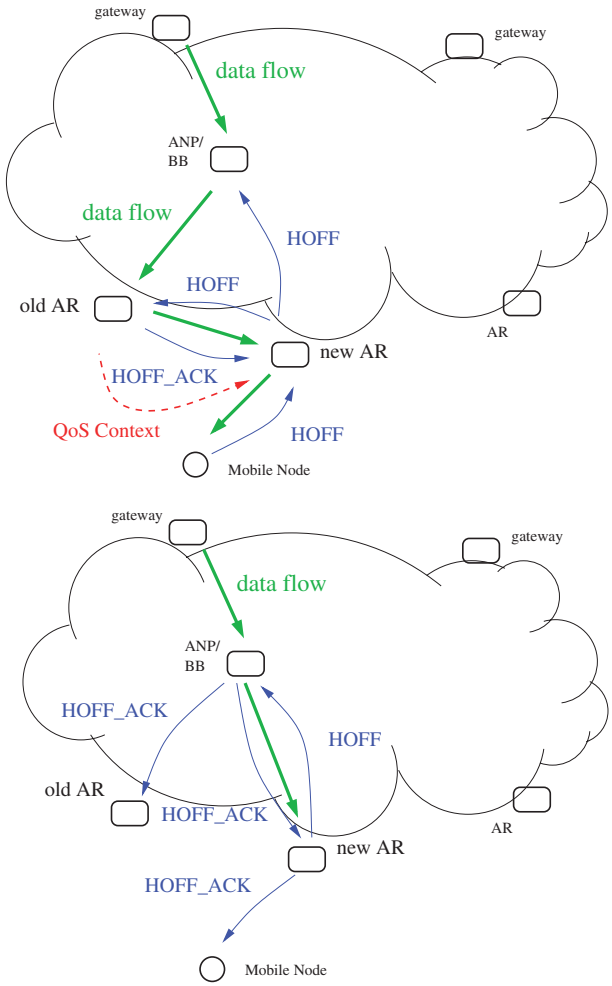

Figure 7. Handover procedure

flow. Then, the AR forwards the HOFF message to the ANP/BB and the old AR. By reception of the HOFF message, the old AR configures an IP-in-IP tunnel, so that the data packets can be oriented to the new location of the MN. Furthermore, by sending a HOFF_ACK message the old AR signals the tunnel creation and the MN context is transferred. The MN context contains, in our case, the DiffServ configurations (filter, marker, shaper, etc.) of the old AR. The QoS context is then enforced in the new AR. Therefore, the HOFF_ACK message has to be received by the new AR, before the creation of the tunnel. Once the ANP has sent the HOFF_ACK message to the old AR, the old AR clears all information related to the MN: DiffServ configurations and the tunnel to the new AR. It may be pointed out that there is no interaction with the BB, thanks to the QoS context transfer between the old and new ARs.

The context transfer, in the handover execution process, consists of the DiffServ configuration information needed in order to mark the flows belonging to the MN in the uplink direction. The information transferred between the ARs is as follows for each active session of the MN: the DiffServ codepoint associated to a flow and the information to identify this flow (source or destination IP addresses and port numbers). This QoS context transfer avoids DiffServ configuration enforcement by the BB at each handover. Nevertheless, if it is considered that the DiffServ class has to be modified after a handover, which can occur when resources are not available at the new AR, the BB has to enforce the DiffServ configuration at the new AR.

DOI: 10.1002/nem

4.4 *Handover preparation*

The handover preparation procedure (Figure 8) is optional. This mechanism is typically used when the MN has time to prepare its handover. In this case, the mobile host sends an HPREP message to its current AR in order to set up a tunnel to the new AR. In addition to the existing signalling, the following issues have to be considered. In the HPREP message, sent by the current AR to the new AR, the QoS context should be included. Once the new AR receives the context, it acknowledges the current AR by sending an HPREP_ACK message, which enables the IP-in-IP tunnel between both ARs. Before establishment of the tunnel, the current AR should reserve resources for it.

Thus, in the case of a handover preparation, tunnelled packets are forwarded by the current AR to the MN and also to the new AR. In order to implement this planned handover mechanism, 'Netfilter' has been used. IP-in-IP data packets coming from the ANP are filtered at the current AR and are directed towards the BCMP user-space program thanks to the Netfilter NF_QUEUE method as shown in Figure 9. The user-space program duplicates these packets. One of the duplicated packets continues its regular processing in the BCMP program and the other packet is encapsulated in an IP-in-IP header and is sent towards the new AR. At the new AR, the data are decapsulated and forwarded to the MN. Thus, in order to implement the planned handover function, part of the code is a simple kernel module.

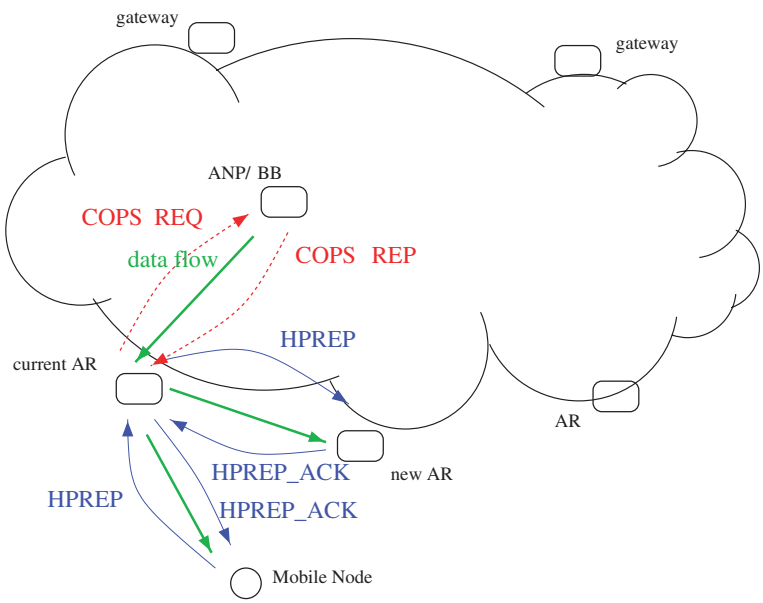

Figure 8. Handover preparation procedure

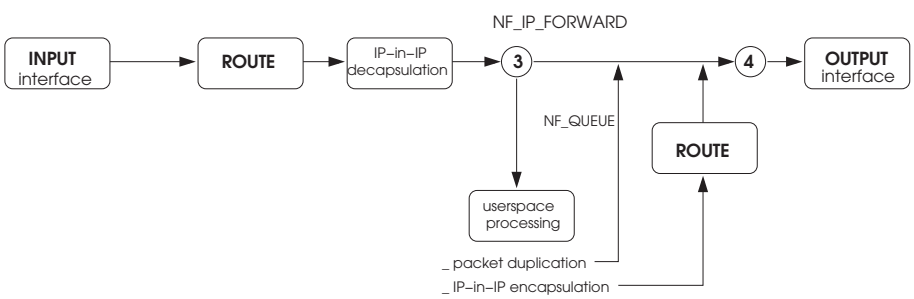

Figure 9. Handover preparation and Netfilter

DOI: 10.1002/nem

5. HANDOVER SIGNALLING ANALYSIS

As a proof of concept, the signalling between the different entities is examined with the protocol analyser Ethereal [13]. In Figure 10 can be seen the Ethereal traces of the PEP initialization phase. These traces show the packets captured at the PEP node (with the IP address 192.168.3.2), which is also an AR as shown in Figure 3.When the PEP is switched on, the PEP establishes a connection to a given PDP by sending a COPS OPN message. Once the CAT message is received at the PEP, a COPS session is established between the PEP and the BB. This session is identified by the client handle object, which can be seen in Figure 11. Once the CAT message has been received, a COPS request is sent, asking for an initial DiffServ edge router configuration. The DiffServ configuration enforced by the BB is contained in the COPS decision object (Figure 11). It can be noticed that this configuration is transported using an SLS template, which is itself transported in the client-specific information object (ClientSI) of the COPS protocol. Figure 12 shows the format of a ClientSI object, corresponding to a C-num = 9 and C-type = 1. Inside the ClientSI COPS object can be found the SLS template object. This object is made of two parts: the encoded provisioning identifier (EPRID) and the encoded provisioning instance (EPRI) data. The PRID and PRI are defined in the PIB.

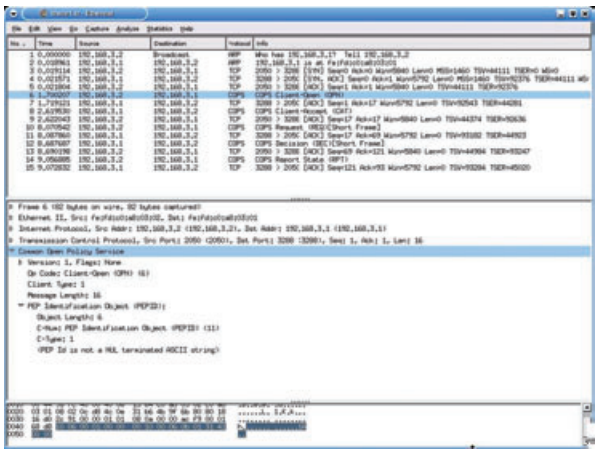

Figure 10. PEP COPS request

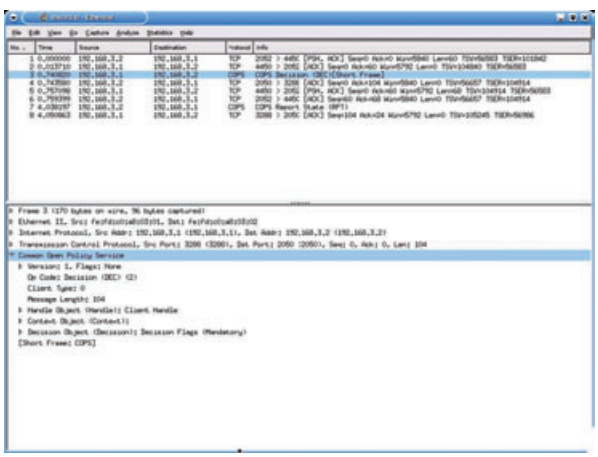

Figure 11. BB decision enforcement

DOI: 10.1002/nem

length of the object	C–num = 9 (ClientSI)	C–type=1 (Signaled)
Object Content (SLS template) = (EPRID + EPRI)		

Figure 12. ClientSI object format

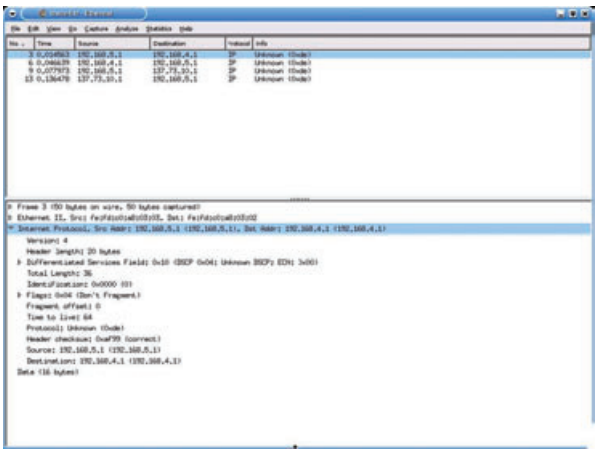

Figure 13. Handover execution

Figure 13 shows the traces of the handover signalling packets, captured at the target AR (with the IP address 192.168.5.1). It can be seen that first a context transfer is done with the current AR. After this context transfer a handover message is sent to the ANP (with the IP address 137.73.10.1), which is co-located with the BB. It can be noticed that the signalling packets have a specific DSCP (0x10) in order not to be lost during a congestion time.

Figure 14 shows the traces of the handover preparation signalling packets, captured at the target AR. It can be seen that the handover preparation signalling consists only of an exchange between the target AR and the current AR. This signalling triggers the set-up of an IP-in-IP tunnel at the current AR and is used in order to transfer QoS context (SLS) from the current AR to the target AR. From this QoS context is extracted the DiffServ configuration, needed at the AR in order to police and mark uplink packets.

Finally, in Figures 15 and 16 can be seen, respectively, the traces of a single ping request after a handover preparation and before the handover execution at the current AR and at the target AR. At the current AR, the first line of the ethereal trace corresponds to the incoming encapsulated ping request from the ANP (with the IP address 137.73.10.1). It can also be seen that there is another ping request packet. This corresponds to the ping request generated locally and sent to the target AR. This ping request is captured at the target AR, as shown in Figure 16. It can be pointed out that the ping reply to the duplicated ping request is going through the current AR, since the MN is still only connected to the current AR. Furthermore, in a real system, the duplicated ping request should not have reached the MN as the link layer connection to the target AR has not been established yet.

6. CONCLUSIONS

Many protocols have been proposed for local IP mobility management; however, there is little literature on QoS-aware IP handover. In this paper, the inter-working of the mobility management and QoS

DOI: 10.1002/nem

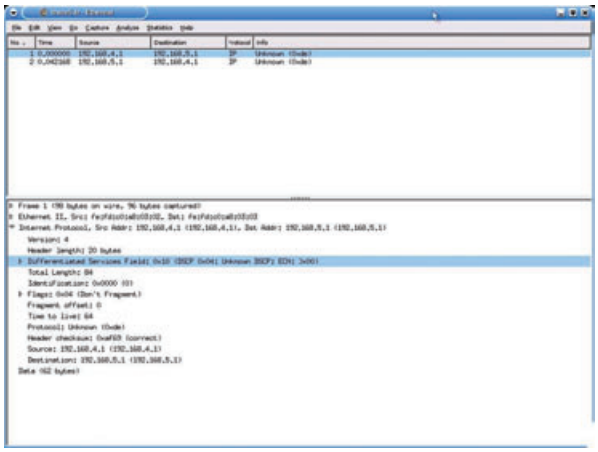

Figure 14. Handover preparation

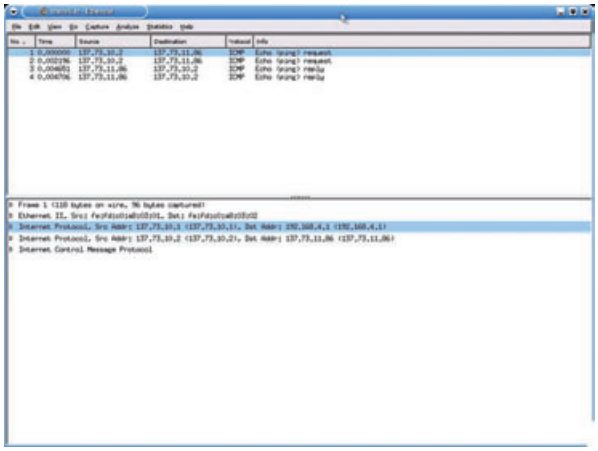

Figure 15. Ping trace at the old AR

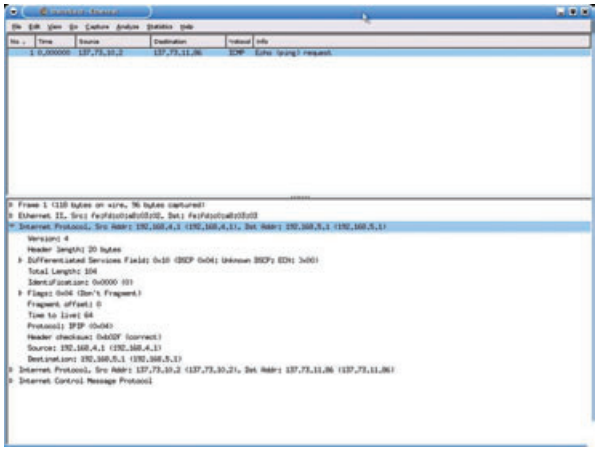

Figure 16. Ping trace at new AR

DOI: 10.1002/nem

entities has been described. Moreover, implementation of the QoS-aware IP handover and the tests using a virtual network testbed has been detailed.

ACKNOWLEDGEMENT

This study was performed under the support of project IST-AROMA (www.aroma-ist.upc.edu), which is funded by the European Community.

REFERENCES

1. Perkins C. Ip mobility support for IPv4. *IETF RFC 3344*, August 2002.
2. Campbell A, Gomez J, Kim S, Wan C. Comparison of IP micromobility protocols. *IEEE Wireless Communications* 2002; February: 72–82.
3. Soliman H, Castelluccia C, El Malki K, Bellier L. Hierarchical mobile IPv6 mobility management (HMIPv6) (draft-ietf-mipshop-hmipv6-04.txt). IETF draft, December 2004.
4. Ceszei K, Georganopoulos N, Turyani Z, Valko A. Evaluation of the BRAIN candidate mobility management protocol. In *IST Mobile Communications Summit*, September 2001; 889–895.
5. IST-1999-10050 BRAIN. Broadband radio access for IP-based networks, deliverable 2.2. [Online]. Available: http://www.ist-mind.org
6. Ramjee R, Varadhan K, Salgarelli L, Thuel S, Shie-Yuan W, Porta TL. Hawaii: a domain-based approach for supporting mobility in wide-area wireless networks. *IEEE/ACM Transactions on Networking* 2002; **10**(3): 396–410.
7. Goderis D, Griffin D, Jacquenet C, Pavlou G. Attributes of a service level specification (SLS) template. *IETF draft*, October 2003.
8. Neilson R, Wheeler J, Reichmeyer F, Hares S. A discussion of bandwidth broker requirements for Internet2 Qbone, 1999. [Online]. Available: http://citeseer.ist.pou.edu/neilson99discussion.html/
9. Zhuang Z, Duan Z, Hou Y. On scalable design of bandwidth brokers. *IEICE Transactions on Communications* 2001; **E84-B**(8): 2011–2025.
10. User-Mode-Linux. [Online]. Available: http://user-mode-linux. sourceforge.net
11. Boukis C, Georganopoulos N, Aghvami H. A hardware implementation of BCMP mobility protocol for IPv6 networks. *IEEE Globecom* 2003; **6**(December): 3083–3087.
12. Jha S, Hassan M. Java implementation of policy-based bandwidth management. *International Journal of Network Management* 2003; **13**(4): 249–258.
13. Ethereal network analyzer. [Online]. Available: http://www.ethereal.com [3 January 2007].

DOI: 10.1002/nem