Telecommun Syst (2008) 37: 49–70
DOI 10.1007/s11235-008-9072-6

A service level model and Internet mobility monitor

Christer Åhlund · Stefan Wallin · Karl Andersson · Robert Brännström

Published online: 20 February 2008

Abstract Mobility is gaining a tremendous interest among Internet users and wireless access networks are increasingly being installed to enable mobile usage. Internet mobility requires solutions to move between access networks with maintained network connectivity. Seamless mobility in turn means that the experience of using a service is unaffected while being mobile. Communication in next generation networks will use multiple access technologies, creating a heterogeneous network environment. Further, roaming between network service providers may take place. To enable mobile nodes to move between access networks within as well as between network service providers with minimal disruption, nodes should be able to maintain multiple active network connections. With the usage of multihomed nodes, seamless mobility can be achieved in already installed infrastructures, not providing mobility support. Mobility in heterogeneous access networks also requires network selections that scale for services. In this article we propose an architecture where application service providers and network service providers define service levels to be used by a mobile node and its user. The user selects a service and the service level from an application service provider. When performing access network selection, information received as part of an application service level will be used to find a network that supports the service required. The performance of available access networks will be monitored and considered when making the decision. Our proposed architecture provides solutions to move flows between interfaces in real-time based on network performance, quality of service signalling to correspondent nodes, and cancellation of flows to give way for more important traffic.

Keywords Mobility management · Service level model · Heterogeneous access networks · Network policies

C. Åhlund (✉) · S. Wallin · K. Andersson · R. Brännström
Division of Mobile Networking & Computing, Luleå University of Technology, SE-931 87 Skellefteå, Sweden
e-mail: christer.ahlund@ltu.se

1 Introduction

Mobility is becoming an important characteristic for Internet access. Services should be available and maintained when roaming between access networks. Examples of such services are ongoing Voice over IP (VoIP) conversations, file download and upload while travelling, etc. Peer-to-Peer (P2P) [2] communication is also a rapidly increasing area. With P2P a new network paradigm is introduced where client computers communicate offering different services compared to the traditional client–server approach. In case of P2P networking client nodes will also operate as servers and this requires that nodes can be located connected to a foreign network.

There are several challenges connected to this issue:

- IP address management;
- Different Access Network Characteristics;
- Service level agreements between a mobile user and service providers;
- Authentication and charging models.

If we assume mobility support, services must continue to operate even in the case of handover. Handover at the network layer requires a new IP address and this needs to be communicated between entities.

Another consideration is the different network characteristics among heterogeneous access networks, see Fig. 1. In such environments network performance will have widely

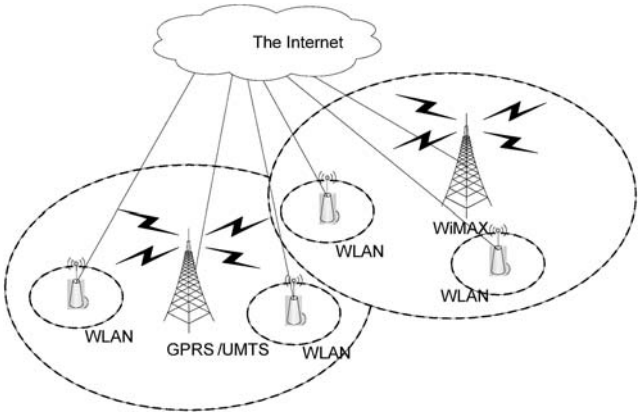


Fig. 1 Heterogeneous wireless access networks

different characteristics and possibly be operated by different network service providers.

When connecting to an access network a Service Level Agreement (SLA) is needed between user and network service provider, as well as application service provider. Such an agreement will define the quality to be expected by the customer of an application as well as the quality of the network connection. A mapping between these two is needed.

Authentication and charging models are important for global roaming to enable mobile users to connect to available networks without pre-subscription. This is however not targeted in this paper.

We need a way to define a formal agreement between a service user and provider. It should specify the capabilities and cost of service (network and higher level). This is the purpose of an SLA. SLAs are by nature contracts or "promises" to be fulfilled. Therefore, it must be possible to measure service level fulfilment at any moment and over time. Characteristics that cannot be measured cannot be part of a SLA. The agreement must be based upon general technical characteristics which are easier to measure, as well as the more challenging user experience based characteristics. The latter requires solutions like probes and involves the underlying problem of probing individual users rather than general network characteristics.

The service levels will host information like Quality of Experience (QoE) which expresses the user perceived quality of a service. QoE in media applications like voice and video is often expressed as the Mean Opinion Score (MOS) [14].

MOS is a well accepted standard for rating the speech quality experienced by the receiver. MOS is a subjective ranking in the scale 1 to 5. In the recommendation G.107, [16], the International Telecommunications Union Technical standards (ITU-T) have developed a computational E-model to describe the objective quality of experience. The E-model produces a rating factor (R value) in the scale 0–100 which can be converted to MOS values according to the voice call

R value	MOS	User perception	Quality rating
90–100	4.34–4.5	Very satisfied	Best
80–90	4.03–4.34	Satisfied	High
70–80	3.60–4.03	Some users dissatisfied	Medium
60–70	3.10–3.60	Many users dissatisfied	Low
50–60	2.58–3.10	Nearly all users dissatisfied	Poor

Fig. 2 Voice call rating

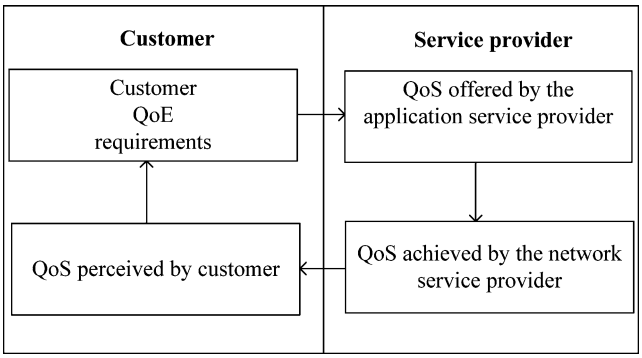


Fig. 3 The relation between the two measures QoS and QoE

rating illustrated in Fig. 2 which is categorized according to ITU-T [15] and [16].

Another important source of QoE parameters is user feedback systems like Customer Care reported problems, trouble ticketing systems, etc. Quality of Service (QoS) included in service levels in turn is a more technical representation of quality and is often related to bandwidth, delay, jitter and Bit-Error-Rate (BER).

The relation between the two measures (QoS and QoE) is illustrated in Fig. 3. The service provider is capable of providing services with a defined QoS, whereas users expect a defined QoE. Our approach to this is the following:

- The service providers publishes service levels in service catalogues;
- Users choose service levels from application service providers rather than formulating them;
- The service levels advertised by the network service provider are selected based on user entered policies and parameters expressed in service levels from the application service provider.

For QoS in IP networks there are two major proposals; Integrated Services (IntServ) and Differentiated Services (DiffServ). The DiffServ approach is the most scalable and accepted solution in today's Internet community. The IP header has a field named Differentiated Service Code Point (DSCP) used by routers and switches (layer 3 switches) in the Internet to differentiate flows with different priorities. Types of network traffic that require high QoS are media-

flows like VoIP and IP-TV, while downloading files (FTP) and emails are less critical. In DiffServ there are 3 major classes; Expedited Forwarding (EF); Assured Forwarding (AF) and Best Effort (BE). Traditionally the Internet has been a BE network but there are ongoing efforts to introduce other QoS classes as well. The EF class is used for time critical applications like VoIP. The AF class is in turn less prioritized than EF and is divided into 4 subclasses.

For QoE fulfilment, application requirements need to be mapped to the more technical QoS parameters, to enable the required network selection and configuration.

This paper describes an architecture that enables inter as well as intra operator mobility in heterogeneous access networks. A framework enabling modelling and monitoring of service levels and services is proposed. The framework also enables communication between applications and network entities. Further, a network layer solution that provides scalable and seamless mobility is presented.

The paper is organized in the following way. Section 2 describes the overall architecture. The service level and service modelling and monitoring framework is presented in Sect. 3. Section 4 describes the proposed network layer solution to enable scalable and seamless mobility. Performance evaluations are presented in Sect. 5. Section 6 describes related work and Sect. 7 concludes the paper and describes future work.

2 The architecture

With the introduction of user mobility in IP networks, the network service provider service level may not be available at the network to which a Mobile Node (MN) is roaming. A typical scenario is a user that connects to the Internet via a WiFi connection in the home, and while running some service disconnects and attaches to a foreign network with a less efficient technology (e.g. UMTS). When the MN requests service forwarding from the home network to the visited network (foreign network), traffic may congest the foreign access network, and be dropped by the access router due to the QoS policing function.

A user must be able to select service levels from an application service provider and then select a network service provider that provides the required service level. That is, there is a need to link between parameters in these two service levels. After an agreement is settled and traffic is communicated, the performance must be monitored to see that it is fulfilled. Due to changes in network conditions and mobility, handover may take place.

In wireless heterogeneous access networks it is not possible to compare physical and datalink layer parameters like the signal-to-noise ratio (SNR) and the BER, etc. between technologies. These parameters differ and are not comparable (e.g. the SNR of a UMTS access point can not be compared to the SNR from a WiFi access point) since there is no correlation to the performance of user traffic at each technology. We therefore make use of network layer characteristics.

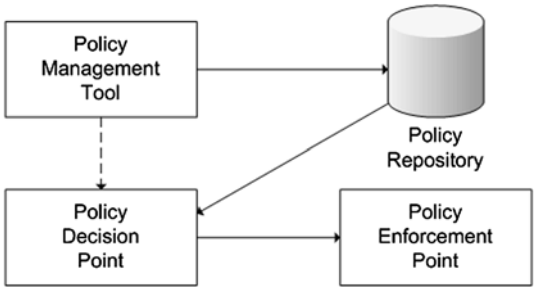


Fig. 4 The policy-based networking architecture

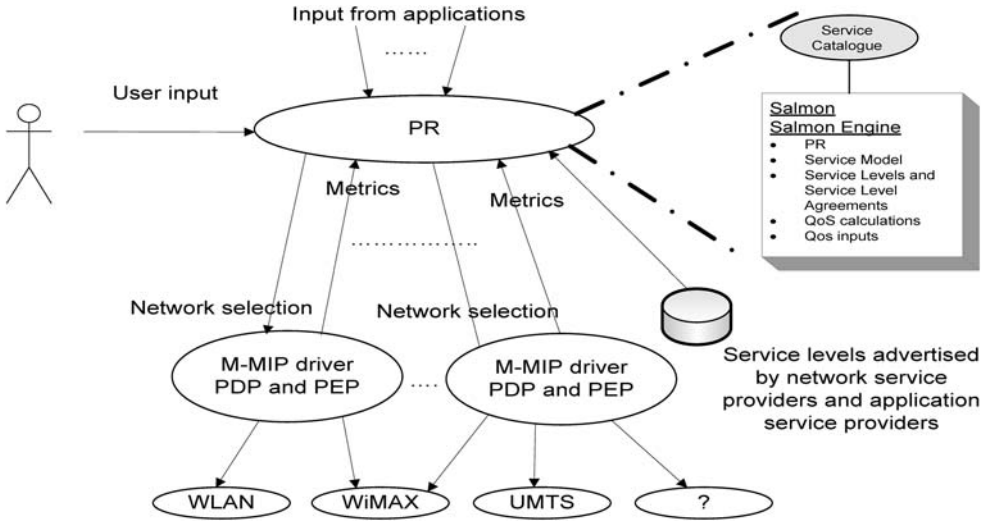
The architecture described in this section monitors access network performance using control messages and this information is used as a parameter when deciding the network service provider.

The performance evaluation of applications data is expected to be managed by the applications themselves. As an example, media communication using the Real Time Protocol (RTP) [30] has sender and receiver reports exchanged using the Real Time Control Protocol (RTCP). This enables an application to monitor the performance and take actions to sustain best possible QoE. When an application discovers that network performance does not scale for the service, the application needs to request a new network connection or adapt accordingly (e.g. by changing codec). Protocols like RTCP only enable measurements of the active flow and cannot be used for performance comparisons between access networks without sending the data in both paths. We enable applications to interact with a policy repository to trigger actions like handover etc.

Our architecture uses policy-based networking [27, 39] which is a popular way of automating network management. Policies comprise a set of rules to administer, manage and control access to network resources and typically to describe configurations, traffic classification, and service levels in SLAs. They often encode high-level goals and requirements for network management and had, at least initially, a network-centric approach. QoS provisioning is one of the most common application areas of the policy-based networking architecture. Four basic elements are defined in the architecture, see Fig. 4: a policy management tool, a policy repository (PR), a policy decision point (PDP), and a policy enforcement point (PEP).

The policy management tool is used to input different policies. It converts high-level policies to low-level, detailed policy descriptions that can be applied to elements in the network. The PR is used to store both high-level policies and low-level policies. The PDP is responsible for interpreting

Fig. 5 The architecture



policies stored in the PR and converting them into a format understood by the PEPs it serves.

In our architecture an application signals the PR when handover needs to take place based on bad performance at the network currently used. This will happen when the communication link does not perform well enough to fulfil the QoE and therefore handover needs to be triggered. Handover is also required when a user leaves the coverage of an active access network. In both cases a new access network needs to be selected. The PR in turn informs the PDP that an action is needed and provides the policy value for available access networks. The PDP will trigger the PEP to select a new network based on these values. Information about network connections is also given by the networking software to the PR to be used in a policy cost calculation.

Figure 5 illustrates how policy-based networking is applied in our architecture. The PR and the policy management tool are implemented based on the Salmon Engine described in Sect. 3 and the PDP and PEP are hosted in a module enabling seamless and scalable mobility named M-MIP described in Sect. 4.

The input to the PR is provided as user policies, requirements of applications and monitored network performance metrics. The PDP and PEP are hosted in the M-MIP driver that also provides a socket Application Programmers Interface (API). The M-MIP driver receives information from the PR and selects the appropriate physical interface for a flow using the PDP and PEP functionality. Different flows may use different physical interfaces. The M-MIP driver monitors the performance by periodically probing each active interface using control messages. The metrics monitored are delay and jitter based on the Round Trip Time (RTT). These metrics are used by the PR to calculate the Relative Network Load (RNL). The RNL metric (described below) benchmarks between technologies, and if a technology has a lower metric compared to another it indicates:

- A shorter distance to the peer based on the delay;
- The path is less loaded since the jitter is low (low delay in buffers and the scheduling is managed in a timely manner in networking components along the path).

Different probing strategies are required for different technologies. Probing access networks with good performance and small cells (e.g. WiFi with steep slopes in cell edges) need to be frequent depending largely on the speed with which users enter and exit cells. For technologies like UMTS, cell coverage is much wider and resources more scarce. In such cases probing may be less frequent. Our policy cost function is defined as:

$$P_{x,y} = \sum_{i=1}^{m} w_i \ln(V_i) + w_{RNL} \ln(RNL_n) \quad (1)$$

where

$$\sum_{i=1}^{m} w_i + w_{RNL} = 1.$$

$P_{x,y}$ is the policy value for access network x and node y and is the weighted sum of normalized parameters and dynamic metrics. The parameters V and the weights w are entered by the user as well as downloaded from service providers. The RNL in turn contains the monitored performance of access technologies. Access network selection decisions are based on this policy cost function.

The choice network interfaces to consider for services are decided by the user. A user defines the importance of different traffic types by expressing the priority and interfaces to use (see Fig. 6).

The example illustrates a scenario where VoIP is considered a high priority service and all available interfaces should be used to keep the conversation going. For the email

Application	Priority
Voice over IP	High
Email	Medium
Some application	Low

Priority	Active interfaces
High	WiFi, UMTS, WiMAX
Medium	WiFi, WiMAX
Low	WiFi

Fig. 6 Binding between applications and interfaces base on priority

application, UMTS communication is considered too expensive and messages in real time are not important. Based on this, emails can be deferred until a WiFi or a WiMax network is present. For low priority traffic only WiFi hotspots will be used. To select the network interface for a flow, a comparison of listed interfaces for an application service is carried out based on the $P_{x,y}$ (see formula (1)). E.g. the VoIP service in Fig. 6 may use WiFi if all access technologies are present and WiMAX (if UMTS and WiMAX is available) depending on the results from the policy cost function.

The RNL metric calculation in formula (2)–(6) is used to consider network performance and is based on a simplified version of the Relative Network Load metric [1]. The metric calculation in Åhlund [1] was developed for infrastructure and ad hoc networks based on the 802.11 technology where mobile nodes are in a single collision domain. The article Åhlund [1] also assumes that gateways sending advertisements are present. The RNL calculation described in this paper instead addresses heterogeneous access networks and does not assume any infrastructure support for mobility like gateways advertising their presence. RTT and jitter in RTT values are calculated on control messages for mobility forming an RNL metric representing a quality value for each access network. RTT and RTT jitter values are access technology independent and are good indicators of congestion in networks and limitations in available bandwidth. Examples can be RTT metrics to an anchor point at the home network when an MN is visiting a foreign network, or RTT metrics to peers enabling P2P connectivity performance indication.

The RTT jitter, being the variation in RTT, is calculated using formulas in RFC 3550 [30]. The formula is adjusted with a variable history window instead of a fixed history window of 16 giving the following formulas:

$$RNL_n = \bar{z}_n + cJ_n \tag{2}$$

$$\bar{z}_n = \frac{1}{h} RTT_n + \frac{h-1}{h} \bar{z}_{n-1} \tag{3}$$

$$RTT_n = R_n - S_n \tag{4}$$

$$\begin{aligned} D_n &= R_n - R_{n-1} - (S_n - S_{n-1}) \\ &= (R_n - S_n) - (R_{n-1} - S_{n-1}) = RTT_n - RTT_{n-1} \end{aligned} \tag{5}$$

$$\mathrm{J}_n = \frac{1}{h} |D_n| + \frac{h-1}{h} J_{n-1} \tag{6}$$

where, S_i and R_i are defined as

S_i = the time of sending binding update message i

R_i = the time of arrival of binding acknowledgement message i

h determines the history window for the weighted average calculations. For example, when $h = 5$, the most recent value will contribute to the calculated $\bar{z}_n$ and J_n values with 20%.

c determines the weight of the RTT in comparison to the RTT jitter value. For example, when $c = 5$, the RTT jitter value is contributing five times more to the RNL metric value than the RTT value does.

The variables $\bar{z}$, D, and J are initialized with the following values:

$$\bar{z}_0 = RTT_0$$
$$D_0 = 0$$
$$J_0 = D_1.$$

The RNL metric is beneficial for its access network independence feature. The fact that no synchronized clocks are needed also favours this solution.

Before any handover decision is finally made, the direction of the vertical handover is determined. Nasser [28] defined an upward vertical handover as roaming to an access network with a larger cell size and lower bandwidth, and a downward vertical handover as roaming to an access network with a smaller cell size and larger bandwidth. Using those definitions, a handover is decided when

$$P_{\text{downward-access-technology}} < P_{\text{upward-access-technology}} - \text{hysteresis}$$

for downward handovers where a hysteresis is used to avoid a ping-pong effect between interfaces. Upward handovers take place when

$$P_{\text{upward-access-technology}} < P_{\text{downward-access-technology}}.$$

This asymmetric decision model is used in order to let handovers to access networks with high but unstable capacities wait until the policy value is significantly lower compared to the old access network with lower capacity but better coverage. When doing handovers to access networks with less capacity but better coverage, the handover decision should be executed immediately in order not to lose the connection. This is especially important at high speeds and steep cell edges. An example of an upward access technology can be UMTS and a downward access technology WiFi.

The link between the network layer performance managed by M-MIP, application requirements and user input is provided by the Salmon Engine (described in the next section) implementing the PR. The calculations presented in this section take place in this PR. The Salmon Engine enables the creation of service levels for both application service providers of services like VoIP, IP-TV, etc., and network services. The service levels advertised by an application service provider include information about network performance requirements like bandwidth, delay, jitter and loss. This information is stored in the MN to be used when connecting to a network service provider.

An automated process is triggered when connecting to a network service provider. The parameters stored for an application service are matched against the service levels that the network service provider advertises. In case of a match, a DiffServ class is given to the MN to be used by packets communicated by the service. The M-MIP driver described in Sect. 4 manages the QoS signalling required.

The architecture depicted in Fig. 5 is hosted in the mobile node. This makes the network selection a fully Mobile Controlled Hand-Over (MCHO) scheme enabling our solution for mobility in already installed heterogeneous access networks. New access network technologies can easily be added since the M-MIP driver makes use of all installed interface technologies in an MN listed to be active.

3 Salmon; modeling and monitoring of service levels and service quality

In order to offer an option for service modeling and service status calculation we are developing a service monitoring language and framework named Salmon Wallin [36]. We use a tailor-made pure functional language for defining services and service levels. This enables us to create services using well understood methods of program construction. The language has two main purposes: first, it defines the structure of the model, and second, it defines the relationships between parameters and determines how they will be computed.

The language requires a special runtime environment, the Salmon Engine. The engine is responsible for marshaling inputs and outputs from the language and evaluating the expressions. External programs can subscribe to value changes and read/update values via the supported API and protocols. In the architecture described in this paper, applications as well as the M-MIP driver will interact with the Salmon Engine.

In the solution described here Salmon is used to implement aspects of a policy based management architecture. The language is used to model services in the service repository, and the engine is used to calculate the service status. Figure 5 illustrates the placement of the Salmon Engine in the architecture. Referring to Fig. 4, the Salmon Engine includes the functionality in the Policy Management Tool and PR. Further, since PR carries out modeling and monitoring it can be viewed as an active PR.

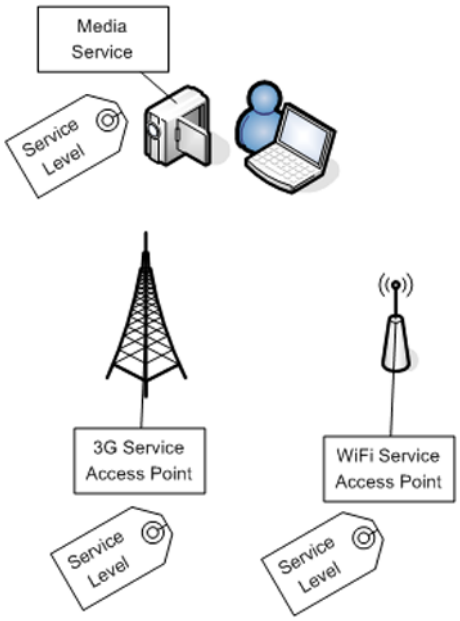


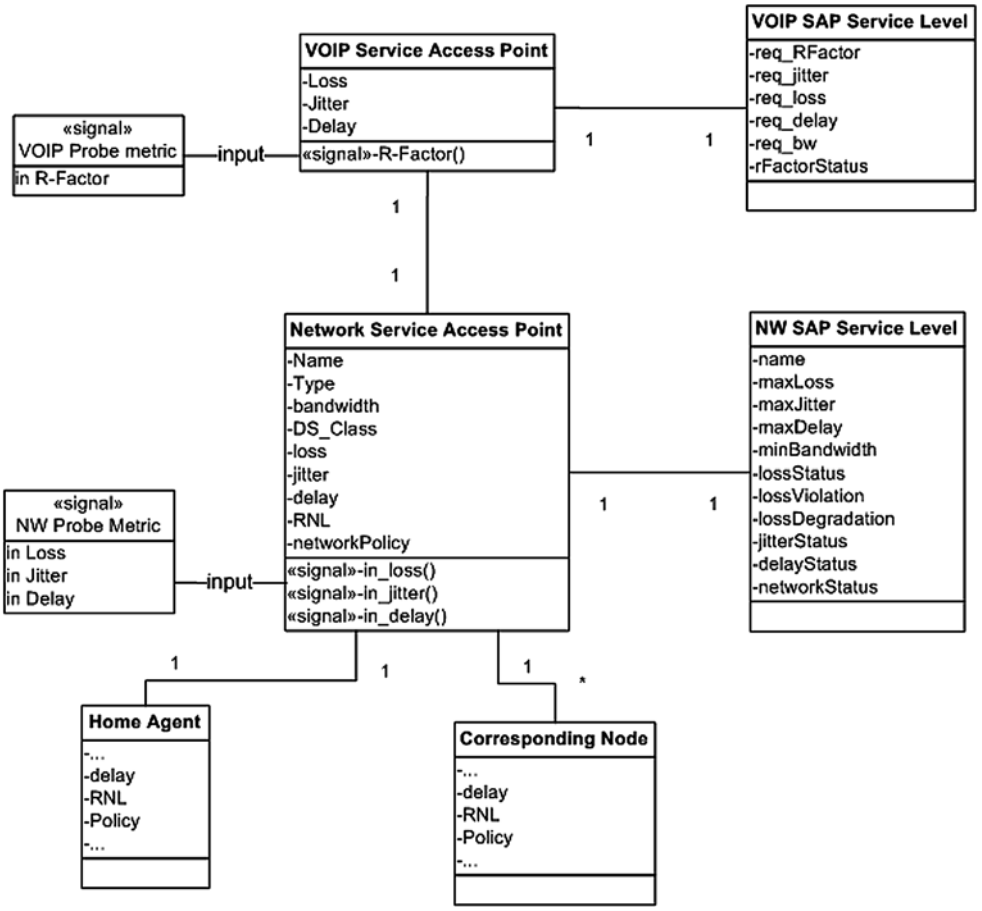
Fig. 7 Conceptual model for VoIP scenario

Although Salmon is focused on modeling and monitoring, it also plays a role in the overall service life cycle. The service is formally defined using the language; the service definition can be published in a catalogue with different service levels; services are instantiated in the engine when they are deployed and configured; the engine monitors the service quality and the SLA fulfillment.

To provide a clear description of the Salmon Engine we make use of a VoIP scenario with different service access points, see Fig. 7. We will go through the following steps:

1. Modeling:
   - The media service is modeled, including QoS calculations, and inputs;
   - Service Access Points, SAPs, for 3G and WIFI are modeled;
   - SLs are defined by the providers.
2. Publishing:
   - The service levels are published in a Service Catalogue.
3. Managing inputs:
   - Source systems for measuring raw QoS metrics are identified and interfaced to the Salmon engine. This is done using probes provided by the M-MIP driver.

Fig. 8 The VoIP model



4. Instantiation:

   - The network service instances are instantiated. The engine will start to collect and calculate network service parameters.

5. Use of Salmon Engine:

   - External programs like a PDP can subscribe to variables in the Salmon Engine and take appropriate actions like hand-over. In our architecture the PDP is located in the M-MIP driver.

A simple Salmon definition for the VoIP service and service level is given below, see Fig. 8. The model defines two service point classes, VoIP and Network, with relevant QoS parameters. Associated with each service access point is a Service Level. The Service Level expresses constraints on the QoS parameters. Probes feed the service instances with input metrics. This is expressed in Salmon as "inputs".

Figure 9 shows the Salmon code used for the SAPs. The code in this figure and the following is not complete in order to enhance clarity.

The `VoipSAP` takes the *R*-factor as input from probes and applications. It is used to calculate the network policy according to formula (1). Finally it aggregates network QoS parameters from the associated network service access point. The `NetworkSAP` has the associated DiffServ class as a property and takes network QoS parameters as inputs. The last part of the `NetworkSAP` expresses the RNL calculation. In the example above we note the following characteristics of the Salmon language:

- Input from probes and other QoS metrics are treated as "input" in the definition. In this example we assume that there are VoIP probes for VoIP related QoS parameters;
- Relationships between service instances are defined by "anchors". The VoIPSAP has an anchor to the bound network service access point. Attributes of the associated instance can be accessed;
- Variables are indexed by time. This allows us to support advanced QoS parameter calculations. QoS parameter calculations are often time-dependent, such as calculating the difference between two samples. A more challenging requirement is that probe data arrives late so that parameters need to be changed retrospectively along with the dependant calculations. Salmon supports continuous as well

```
class VoipSAP
  input rFactor;
  anchor networkSAP;
  loss = networkSAP.loss;
  jitter = networkSAP.jitter;
  delay = networkSAP.delay;
  bandwidth  =
        networkSAP.bandwidth;
```

```
class NetworkSAP
 properties name, type, DS_Class;
 input in_loss, in_jitter,
       in_delay, in_bandwidth;

 loss = in_loss;
 jitter = in_jitter;
 delay = in_delay;
 bandwidth = in_bandwidth;

// RNL calculations
 RNL = z + c * j;
 z = 1/h * in_delay + (h-1)/h * z@prev;
D = in_delay- in_delay@-1;
 J = 1/h * abs D+ (h-1)/h * J@prev;
let h = 5;

// Policy according to (1)
networkPolicy = … RNL…
```

Fig. 9 Salmon definition of Service Access Points

```
class VoipSAPSL
  properties req_RFactor, req_loss,req_jitter, req_delay, req_bandwidth;
  anchor voipSAP;
  rFactorStatus = linearThreshold voipSAP.rFactor 0 req_RFactor;
```

Fig. 10 Salmon definition of a VoIP Service Level

as discrete time domains. The example above uses discrete references like *in_delay@prev* which refers to the previous value. Continuous time references are expressed as *in_delay@NOW -1m*, which requests the value one minute ago.

The Salmon definition in Fig. 10 illustrates the definition of a Service Level on the VoIP service based on the *R*-factor.

The *VoipSAPSL* class expresses requirements on the network layer in the form of required loss, jitter, delay and bandwidth.

A Service Level is always associated with its Service Instance; in this case the VoIP service level is associated to the VoIP Service Instance using the anchor *voipSAP*.

A fundamental part of service levels are thresholds based on vital QoS calculations. Different levels of thresholds identify degradations and violations. Salmon has implicit support for thresholds. The above example illustrates a linear threshold comparing the current *R*-factor, *voipSAP.rFactor*, with the established threshold in the Service Level, *req_RFactor*.

The network SAP Service Level definition illustrated in Fig. 11 further illustrates some Salmon capabilities. We define the network service degraded when the loss status is above 80%, *lossDegradation = lossStatus@NOW > 0.8*. The *lossStatus* is a linear threshold between 0 and the max loss according to the service level.

The last statement in the network service access point shows a typical abstraction in service modeling. In general our task is to determine the current (and historic), status of the network. Networks cannot only be judged by a set of low-level QoS parameters. Rather, we need to provide general high-level status attributes on “health”. In this specific case we apply a worst-case evaluation on the available status calculations.

Based on the abstract Service Levels classes above, we can define concrete Service Levels with different thresholds

```
class NetworkSAPSL
  anchor networkSAP;
  properties name, maxLoss, maxJitter, maxDelay, minBandwidth;
  lossStatus = linearThreshold networkSAP.loss 0 maxLoss;
  lossViolation = lossStatus networkSAP.loss > maxLoss;
  lossDegradation = lossStatus > 0.8;
  jitterStatus = linearThreshold networkSAP.jitter 0 maxJitter;
  delayStatus = linearThreshold networkSAP.delay 0 maxDelay;
  bandwidthStatus = linearThreshold networkSAP.bandwidth minBandwidth 0
  networkStatus = worst
        [lossStatus, jitterStatus, delayStatus, bandwidthStatus];
```

Fig. 11 Network SAP Service Level definition

```
def MOS4VoipSAPSL = VoipSAPSL(req_RFactor => 80,
                              req_loss => 0,1, req_jitter => 5,
                              req_delay =>150, req_bw =>20)

def MOS4VoipNwSAPSL = NetworkSAPSL(maxLoss => 0.1, maxJitter => 5,
                                   maxDelay => 150, minBandwidth => 20)
```

Fig. 12 Instantiation of Network SAP Service Levels

based on QoS parameters, see Fig. 12. We define a service level for VoIP with an *R*-factor of 80%. The status of this service level can be monitored for compliance. A network with max loss of 0.1%, max jitter 5 ms, max delay 150 ms and min bandwidth of 20 kbps will enable a MOS 4 perceived quality which is equivalent to *R*-factor 80%.

The specific Service Levels are published in a service catalogue. This is an interface to the Salmon Engine which selects available service levels and presents them. In the given example the *R*-factor depicts the *VoIP* service level whereas loss, jitter, delay and bandwidth define the *network* Service Levels. The instantiation is shown in Fig. 13. When the user has selected the network access and service levels, the model is completed with the associations shown in Figs. 14 and 15.

The Salmon engine constantly calculates all the attributes of the service model and handles inputs for the QoS parameters. The user can monitor the service quality in real-time and historically.

The PDP can now check the policy cost including the network performance, application requirements and user status preferences and try to find a better network access when the VoIP service quality is too low.

The scenario for a hand-over is outlined below using the Salmon API:

1. The PDP subscribes to `voipSLA.rFactorStatus`.
2. If the `rFactorStatus` is poor, look for better network access by retrieving the `networkPolicy` attribute of the available network service access points.
3. The PDP disconnects from the poor network and connects to the selected network. For example:
   - connect `myGoodVoipSAP.networkSAP myGoodWiFiNW`;
   - perform the actual handover by issuing requests to the PEP.
4. The QoS class that will be used by the VoIP flow is obtained by M-MIP via the command *myGoodWiFiNW.DS_Class*.

The description of the Salmon Engine in this section used a VoIP service to illustrate its capabilities. However any type of services (applications and network) can be modelled using Salmon. Referring to formula (1) we only used RNL for our policy cost function in this section: $P_{x,y} = \ln(RNL_n)$. Salmon can easily be used to calculate and manage more complex policy calculation as well.

```
//Instantiate access networks.
create NetworkSAP myGood3GNW(name => "good3GOperator", type => "3G"
                                    DS_Class => "EF")
create NetworkSAP myGoodWiFiNW(name => "goodWiFiOperator",
                                     type => "WiFi", DS_Class => "AF3" )

// instantiate voip service
create VoipSAP myGoodVoipSAP

// Instantiate SLAs.
create MOS4VoipSAPSL voipSLA
create MOS4VoipNwSAPSL nwSLA
```

Fig. 13 Instantiation of services and SLAs

```
  // bind the voip service to use 3G
connect myGoodVoipSAP.networkSAP myGood3GNW

  // bind the voip and network SLAs
connect voipSLA.voipSAP myGoodVoipSAP
connect nwSLA.networkSAP myGood3GNW
```

Fig. 14 Binding service instances to SLAs

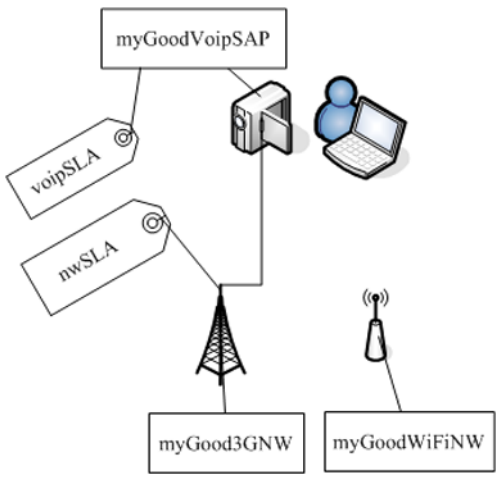


Fig. 15 Bound service instances

4 Seamless and scalable Internet mobility management

An important step towards mobility in IP networks is the introduction of the Mobile IP (MIP) [29]. MIP enables users to connect to different networks (foreign networks) and still maintain the same accessibility as when connected to the home network. With MIP the design of the IP and the IP-address identifying both a host and its location can be sustained. In the MIPv6 [18] standard only one IP address can be registered and all traffic destined for the mobile node will be sent to this address. In a mobile scenario the services accessed at one network may not be usable when connecting to another network. In such cases it is important to find a network connection (network access service) that scales for the service (application). It is possible that the service may have to be adapted to current networking conditions. If none of these actions can take place and a number of services are used in parallel, a service may have to be interrupted or ended to make way for a more important service.

4.1 An introduction to mobile IP

This section describes the MIPv6 architecture that incorporates a home agent (HA), a correspondent node (CN), and the mobile node (MN). An MN connected to the home network will operate according to normal IP network operations, without using the MIP. When an MN connects to a foreign network it will register its new location with the HA based on the address used by the MN in the foreign network. This address is called a care-of address and there are two options for an MN to receive an address: first, via stateless auto-configuration, based on the neighbour discovery protocol (NDP) [5] and second, via statefull process using a DHCP server.

The registration sent to the HA by an MN connecting to a foreign network will create a binding in the HA, between the home address and the care-of address. This is labelled as a binding update and a binding acknowledgement is returned in response.

When packets for the MN are received at the home network, the HA will forward the packets to the care-of address using tunnelling. A tunnel encapsulates the received packet for an MN as a payload in a new packet with an IP header having the care-of address as the destination and the HA as the source. When the packet arrives at the MN at the foreign network, the packet will be decapsulated by the networking software and handed to upper layers.

The interception of packets by the HA for an MN is based on the NDP. A router connected to a network receiving a packet for a destination in the network, or a source in the same network as the destination, will translate the IP address to a Media Access Control (MAC) address. The MAC address is used to send a frame containing the packet the last hop to its destination. The NDP is used to send a request for a MAC address containing the IP address. The host configured with the IP address will respond with its MAC address. When an MN has registered a care-of address at the HA, the HA will respond to the NDP requesting a MAC address for the MN's IP address. The returned MAC address will be the MAC address of the HA.

All hosts as well as routers connected to a network maintain caches for the binding between IP and MAC addresses. To prevent caches from keeping obsolete bindings when an MN registers at a foreign network, the HA sends a gratuitous Neighbour Advertisement message to update these caches with the binding between the MN's IP address and the HA's MAC address.

A packet sent to the CN will use the MN's care-of address as the source, and the home address will be added in the home address destination option. Since the addresses are topology-correct, ingress filtering is avoided. The CN receiving the packet will put the address in the home address destination option as the source address before handing it to the transport layer.

The routing created by MIP is referred to as triangular routing where packets from a CN are sent to the HA.

The HA tunnels packets to the MN, and the MN sends them directly from its current location to the CN, making a triangle (if reverse tunnelling is also used it is named quadrilateral routing). To optimize routing between the MN and a CN, route optimization is used.

A CN sending packets to an MN is informed by the MN about the care-of address used by the MN. When a CN receives a binding update it will start to send packets directly to the MN using the care-of address as the destination address. Support for route optimization is built into the IPv6. The CN will use the routing header in IPv6, where the destination of the packet is the MN's care-of address and the address in the routing header is the MN's home address. When the MN receives such a packet, the destination field will be updated with MN's home address before handing the packet to the transport layer.

To secure route optimization in MIPv6 the return routability procedure is used. For this, four messages are sent; Home Test Init (HoTI), Care-of Test Init (CoTI), Home Test (HoT) and Care-of Test (CoT). Before the MN sends a binding update to the CN it sends a HoTI message to the CN through the HA. A CoTI message is also sent directly to the CN according to IP routing. When receiving these messages the CN responds with the HoT and CoT messages, where HoT is sent through the MN's HA and the CoT message is sent directly to the MN's care-of address. The MN derives a binding management key from the information in the HoT and CoT messages. After this, the binding update will be sent to the CN. The CN will derive the binding management key from the information in the binding update.

The return routability procedure verifies that the MN is reachable through both its home address and its care-of address. To secure the information exchanged in the return routability procedure, IPSec [19] can be used between the MN and it's HA for the HoTI and HoT messages. A malicious node has to intercept both HoT and CoT messages to create the binding management key.

Return routability is required each time the MN changes care-of address. As long as the same care-of address is used the same binding management key is valid.

The Mobile IP solution is attractive since it enables mobility with the most widely used protocols at the transport layer, the Transport Control Protocol (TCP) and the User Datagram Protocol (UDP). Protocols above the network layer are unaware of network mobility. One problem with MIP is the registration time when moving between networks. MIP will probably be most used with MNs connecting wirelessly and this may cause problems because of rapidly changing conditions in the wireless network. An MN switching between APs connected to different networks will require a new registration each time. The time it takes doing

handover may cause UDP packets to be dropped and TCP flows to break.

4.2 Protocol and algorithms

This section describes the algorithms and control messages at the network layer used in our approach. The algorithms only highlight the most important functionality and are not full descriptions. The control messages in turn only host the fields of interest in an algorithm.

We use the MIP as the underlying architecture and make extensions that enable performance comparison between network services using the RNL metric and seamless mobility in already installed network infrastructures. We also enable flow mobility where flows can be communicated via different access network technologies depending on the performance. Further, ongoing flows can be pruned to give way for others. The algorithms described in this section are hosted in the M-MIP driver in the MN depicted in Fig. 5 and the HA at the home network. As earlier stated the architecture is a MCHO mechanism so the PR, PEP and PDP are all hosted in the MN. The variables used are shown in Table 1.

$N_{\text{listed-interfaces}}$ consist of all access networks (e.g. WiFi, WiMax and UMTS) that should be used if they are available. N_{foreign} is the access networks registered at the HA and in the case of route optimization at the CNs. This means that the set N_{foreign} is a subset of $N_{\text{listed-interfaces}}$, that is $N_{\text{foreign}} \subseteq N_{\text{listed-interfaces}}$. The set N_{cn} contains all CNs able of performing route optimization. Bindings between home addresses and care-of addresses and related information are hosted in the set B_{mn}. The set F_{mn} contains the flags used in control messages. The array M_{rtt} consists of RTT measurements between the MN and the HA and CNs. When multiple network interfaces are used in parallel, different care-of addresses may be selected for the HA and CNs, and this information is stored in the array $M_{\text{defInterface}}$. References to the *NetworkSAPs* for networks that are listed to be used are stored in M_{nwSAP}. Finally, the array T_{mn} consists of the tunnels used. For non M-MIP aware CNs all traffic will be sent via the home network and the HA.

The processing in an MN when a new access network is discovered is shown in Algorithm 1 and the binding update message is depicted in Fig. 16. The extensions to the MIPv6 standard are the M, S, D and P flags as well as the DiffServ field. The M-flag informs the HA to keep previous bindings in the binding table for the home address (in the MIPv6 standard a new care-of address registration will erase a former binding). The S-flag is used to inform the HA of which of the registered care-of addresses to use to communicate with the MN. The selection is based on the policy cost function (formula (1)) managed by the Salmon Engine where the RNL metric is calculated according to formulas (2)–(6) based on RTT measurements. The RTT measure is the time between a sent binding update and the time a binding acknowledgement is received.

Traffic destined for the MN at the home network may contain a DiffServ class that does not conform to the access network that the MN currently connects to. Based on the SLA between the network service provider and the MN's user, a different DiffServ class may be required and the HA and possibly some CNs may need to be informed. By adding the capability to include QoS signalling in the binding update, an MN will be able to inform the HA and CNs of the QoS class(es) to use when forwarding/sending packets to the MN. The DiffServ information (DSCP bits) is obtained from the Salmon Engine via the *NetworkSAP*.

When the MN informs its HA of the QoS class to use, it inserts the DSCP bits in the DiffServ field and adds the D-flag. When receiving such a message the HA stores the DSCP bits from the DiffServ field in the binding cache. When packets destined for the MN (connected to a foreign network) appear at the home network, the HA intercepts this traffic and adds an outer IP header (creating a tunnel) with the DSCP bits from the binding table. This means that there is a transformation from the QoS class between the CN and the MN's home network to the QoS class between the home network and the foreign network that was decided as part of the SLA agreement with the network service provider. In the

Table 1

var	
N_{foreign}	: set of registered foreign networks;
$N_{\text{listed-interfaces}}$	: set of interfaces listed to be active by the user;
N_{cn}	: set of correspondent nodes using route optimization;
B_{mn}	: set of bindings;
F_{mn}	: set of flags;
M_{rtt}	: array of RTT measurements;
$M_{\text{defInterface}}$	: array of default interfaces used for the HA and CNs;
T_{mn}	: array of tunnels;
M_{nwSAP}	: array of references to NetworkSAPs;

```
Processing new access network found (new-access-network, care-of-address): begin
  if new-access-network ∈ N_listed-interfaces then begin
    N_foreign := N_foreign ∪ {new-access-network};
    if (dscp := M_nwSAP [new-access-network].ds_class) ≠ ε then
      F_mn := F_mn ∪ {d-flag};
    if |N_foreign| > 1 then
      F_mn := F_mn ∪ {m-flag};
    send <binding update, home-address, F_mn, dscp> to ha via new-access-network;
    M_rtt [ha, new-access-network] := clock;
    forall cn ∈ N_cn then begin
      send <binding update, home-address, F_mn, dscp> to cn via new-access-network;
      M_rtt [cn, new-access-network] := clock
    end
  end
end
```

Algorithm 1 The processing in an MN when a new access network is discovered

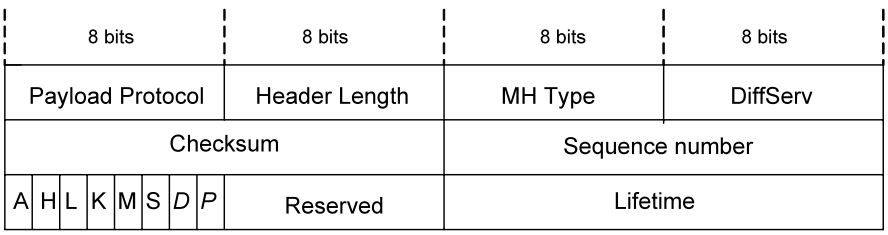


Fig. 16 Binding update message

case of M-MIP aware CNs the same procedure takes place with binding updates sent for route optimization.

When an interface discovers an access network configuration takes place and the interface becomes active. The interface is then checked against the active services (see Algorithm 1) to see if the interface is listed as an alternative (see Fig. 6). When registering a new care-of address at the HA all M-MIP aware CNs are also updated with the new address. The time-stamp when the binding update is sent is stored to calculate the RTT when the binding acknowledge is received. If there is a previously registered care-of address for the MN, the M-flag is set by the operation $|N_{foreign}| > 1$. The S-flag is only used in the case of multiple bindings. In the case that only one binding is present it will be used without the S-flag in the binding update. No S-flag is used in Algorithm 1 since the new access network first needs to be evaluated. If the DSCP bits are available the D-flag is included in the binding update and the DiffServ field is assigned the QoS class.

When a binding update is received at the HA the processing described in Algorithm 2 takes place. The set B_{mn} is the binding cache that hosts all bindings between a home address and care-of address (one or more). The content of B_{mn} is shown in Fig. 17. The home address "3ffe::a:b:c:d" is bound to three care-of addresses. The first row is the care-of address used for all flows (except two) and can be seen as a default care-of address (interface) for the MN. The other two rows are bindings for specific flows. For example, the TCP (protocol number 6) traffic at port 6935 will use the care-of address "3ffc::1:6:a:b:c:a" instead.

Our architecture enables mobility with flow granularity. A single flow can be sent to one care-of address while the rest of the flows are forwarded to another. This takes place in the MN, HA and CNs (in case of route optimization) where the flow identifier (protocol and port) are monitored and used to identify the flow and the care-of address. To enable the MN to inform the HA and CNs of a flow binding we add a flow mobility option header to the binding update, including the protocol and port number (see Fig. 18). However other types of flow identification could be used as well. To add multiple flow bindings in one binding update, a number of flow mobility option headers can be added.

The binding update message in Algorithm 2 has the fields *dscp* and *flow-id* as options so their presence in the

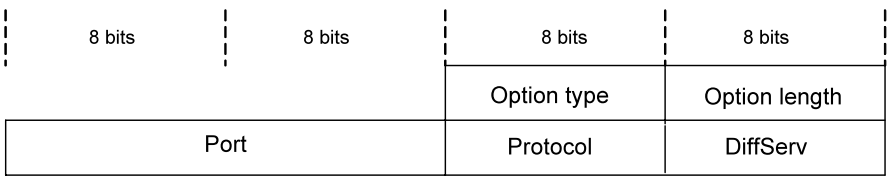
```
Processing binding update: begin
  receive <binding update, home-address, F_mn, dscp, flow-id >  from mn at care-of address;
  if p-flag ∈ F_mn then begin
    binding ∈ {{x, y , z , w} : {x, y, z, w} ∈ B_mn ∧ x=home-address ∧ y = care-of-address ∧ z = flow-id};
    if flow-id ≠ ε then
      send <ICMP destination unreachable, port_unreachable> to cn
    else
      send <ICMP destination unreachable, address_unreachable> to cn;
    B_mn:= B_mn \ binding
  end
  else begin
    if  {home-address, care-of-address, flow-id} ∉ B_mn then
      if d-flag ∈ F_mn then
        B_mn := B_mn ∪ {home-address, care-of-address, flow-id, dscp}
      else
        B_mn := B_mn ∪ {home-address, care-of-address, flow-id, ε};
    if ¬(m-flag ∈ F_mn) then
      forall binding ∈ {{x, y , z, w} : {{x, y , z, w} ∈ B_mn ∧ x=home-address ∧ y ≠ care-of-address ∧ z ≠ flow-id } do
        B_mn:= B_mn \ binding;
    if flow-id ≠ ε then
      T_mn [home-address, flow-id]:= care-of-address
    else if  s-flag ∈ F_mn then
      T_mn [home-address, ε]:= care-of-address
  end
  send <binding acknowledgement> to care-of-address
end
```

Algorithm 2 The processing of registration requests in the HA

Fig. 17 Binding cache

Home address	Care-of address	Protocol	Port	DSCP	Lifetime
3ffe::a:b:c:d	3ffc::1:5:a:b:c:d	−1	−1	BE	150
3ffe::a:b:c:d	3ffc::1:6:a:b:c:a	6	6935	EF	200
3ffe::a:b:c:d	3ffc::1:a:a:b:c:d	17	7830	AF2	150

Fig. 18 Mobility option header



message is optional. When searching for an entry in the binding cache B_{mn} the flow identifier is present as search criteria. If a flow identifier is not present in the binding update the search is carried out without a flow identifier (flow-id $= \varepsilon$). This e.g. means that the expression "binding $\in$ $\{\{x, y, z, w\} : \{x, y, z, w\} \in B_{mn} \wedge x =$ home-address $\wedge\ y =$ care-of-address $\wedge\ z =$ flow-id$\}$;" will return a binding both for a specific flow (e.g. row two in Fig. 17), and in the case

```
Processing time-trigged registration request (access-network) : begin
  if (dscp := M_nwSAP [access-network].ds_class) } ) ≠ ε then
     F_mn := F_mn ∪ {d-flag};
  if |N_foreign| > 1 then
    F_mn := F_mn ∪ {m-flag};
  send <binding update, home-address, F_mn, dscp> to ha via access-network;
  M_rtt [ha, access-network] := clock;
  forall cn ∈ N_cn then begin
    send <binding update, home-address, F_mn, dscp> to cn via access-network;
    M_rtt [cn, access-network] := clock
  end
end
```

Algorithm 3 The processing in an MN for periodical updates

```
Processing binding acknowledgement : begin
  receive < binding acknowledgement> from x via access-network;
  M_rtt [x, access-network] := clock - M_rtt [x, access-network];
  M_nwSAP [access-network].in_delay[x] := M_rtt [x, access-network]
end
```

Algorithm 4 The processing in an MN when a binding acknowledge is received

that flow-id $= \varepsilon$, the default binding for the home address is returned (e.g. row one in Fig. 17).

If the P-flag is present in the binding update, the binding will be erased and the CN will receive an ICMP destination unreachable. In the case that a flow identifier is attached the binding update in a mobility option header (flow-id $\neq \varepsilon$ in Algorithm 2) it is a flow binding and the destination unreachable message is sent with the code *port unreachable*. Without the flow identifier in the binding update the destination unreachable message is sent with the code *address unreachable* meaning that all traffic from the CN will be pruned.

Without the P-flag a new binding will be inserted or an existing binding refreshed. If the D-flag is included the DSCP bits are inserted into the binding cache. In the case that the binding update is sent without the M-flag all bindings except the one just received are erased. If the S-flag is present, this will be the default care-of address used for traffic between the HA and the MN. If a flow identifier is present a tunnel is created for the specific flow, otherwise it will be the tunnel for all flows without a specific binding. Finally, a binding acknowledgement is returned to the care-of address registered.

A binding update can be sent for only one flow and no default selection will then be present in the binding cache. A scenario for this can be that a VoIP flow will be forwarded to the foreign network while the rest of the traffic like IP-TV etc. will stay in the home network. The HA functionality will in such case have to be hosted in the node being the final destination for the traffic in the home network and the MN must be aware (pre-configured) of the flow identifier used for that traffic.

In the case of route optimization where binding updates are sent from the MN to a CN the processing in Algorithm 2 takes place except for the ICMP messages since the CN itself is the destination of the message.

Binding updates messages are also sent in a timely manner where a timer event triggers the processing (see Algorithm 3). In our case the events will depend on the access technology. For technologies with small cells and steep cell edges like WiFi, binding updates will be triggered more often compared to the UMTS and WiMax technology. The speed of movement is also considered.

When a binding acknowledgement is received the processing in Algorithm 4 is executed in the MN. First, the difference between the time the binding update was sent and

```
Processing network status notification (access-network, x, flow-id): begin
  new-access-network := argmin_y{M_nwSAP [y].networkPolicy[x] : y ∈ N_foreign ∧ y ≠ access-network ∧
                        M_nwSAP[access-network].networkPolicy[x] > M_nwSAP [y].networkPolicy[x] + hysteresis_access-network→y};

  if new-access-network ≠ ε then begin
    if (dscp := M_nwSAP [new-access-network].ds_class)) ≠ ε then
      F_mn := F_mn ∪ {d-flag};
    if access-network = M_defInterface[x] ∧ flow-id = ε then begin
      M_defInterface[x] := new-access-network;
      F_mn := F_mn ∪ {s-flag};
      send <binding update, home-address, F_mn, dscp> to x via new-access-network
    end
    else
      send <binding update, home-address, F_mn, dscp, flow-id> to x via new-access-network;
    M_rtt [cn, access-network] := clock
  end
end
```

Algorithm 5 The processing in an MN when a notification is received from the Salmon Engine

the time the binding acknowledgement was received is calculated. This time will include the transmission time from the MN to the HA or CN, the processing time in the receiver of the binding update and the transmission time of the binding acknowledgement as well as a processing time in the MN. Since this method is used to decide among multiple access technologies to the same HA or CN, the difference between times will be reflected by the network transition times if we assume computation times in a node to be equal each time processing takes place. After the RTT calculation the instance of the NetworkSAP class is provided with the measurement so that a new policy cost value can be calculated.

When a notification arrives from the Salmon Engine at the PDP hosted in the M-MIP driver, the `networkPolicy` attribute will contain policy cost values and these will be compared to select the access network. The input is the network used and the HA or CN that are affected by bad performance. This can take place for individual flows or all traffic communicated via the HA and for CNs $\in N_{ch}$. Algorithm 5 illustrates the processing to select the access network for a flow to the HA and CNs. The interfaces having the lowest policy cost value is selected.

The hysteresis is used to avoid a ping-pong effect between interfaces in the case that two interfaces have policy metrics that are close in value. If a new interface is selected, a binding update will be sent at once to trigger the update.

When a binding update is sent directly to CNs in case of route optimization, the CN stores the information in the binding cache, just as the HA does. The difference is the way in which the CN sends packets to the MN. In the case that the HA forwards packets, a tunnel is used. If packets are sent directly from the CN to the MN, the routing option header is used. Two options are possible in this scenario; the first is that only the DSCP field from the binding table is added to the IP header; the second is that the DSCP field that should have been used without considering the DSCP field in the binding update message is added to the routing header as well. The second option requires an extension to the type-2 routing header. This enables the MN to discover the QoS class that should have been used without the QoS transformation.

For traffic sent by the MN two situations are valid when connected to a foreign network. If traffic is sent via the HA the outer IP header contains the QoS class used between the MN and the HA, while the inner IP header contains the QoS class to be used between the home network and the CN. If the MN sends packets directly to the CN, the IP header contains the QoS class between the MN and the CN. A QoS destination option can also be added to the IP header to inform the CN about the QoS class that should have been used if the flow was sent from the MN's home network. This QoS class is only used to inform the CN.

When a packet is received at the MN the processing illustrated in Algorithm 6 takes place. First, a control is made to see if an update of the CN is needed. This will happen if the packet arrives on an interface different from

```
Processing receive packet: begin
  receive <packet, home-address, flow-id> from x for cn via access-network;
  care-of-address := { y : {x, y, z, w} ∈ B_mn ∧ x = home-address ∧ z = flow-id };
  if care-of-address ≠ access-network then begin
    if (dscp := { w : {x, y, z, w} ∈ B_mn ∧ x = home-address ∧ y = care-of-address ∧ z = flow-id }) ≠ ε then
      F_mn := F_mn ∪ {d-flag};
    if |N_foreign| > 1 then
      F_mn := F_mn ∪ {m-flag};
    if { z : {x, y, z, w} ∈ B_mn ∧ x = home-address ∧ y = care-of-address} = ε then
      send <binding update, home-address, F_mn, dscp> to x via interface using care-of-address
    else
      send <binding update, home-address, F_mn, dscp, flow-id> to x via interface using care-of-address;
    M_rtt [x, care-of-address] := clock
  end
  packet-to-upper-layer(packet, x, flow-id)
end
```

Algorithm 6 The processing in an MN when a user data packet is received

```
Processing send packet: begin
  packet-from-upper-layer(packet, home-address, cn, flow-id, dscp-from-upper-layer);
  care-of-address := { y : {x, y, z, w} ∈ B_mn ∧ x = home-address ∧ z = flow-id };
  if care-of-address ≠ ε then begin
    if (dscp := { w : {x, y, z, w} ∈ B_mn ∧ x = home-address ∧ y = care-of-address ∧ z = flow-id }) ≠ ε then
      send<packet, home-address, flow-id> to cn via interface using care-of-address with QoS class dscp
    else
      send<packet, home-address, flow-id> to cn via interface using care-of-address with QoS class dscp-from-upper-layer
  end
  else begin
    if (dscp := { w : {x, y, z, w} ∈ B_mn ∧ x = home-address ∧ y = M_defInterface[cn] ∧ z = flow-id }) ≠ ε then
      send<packet, home-address, flow-id> to cn via M_defInterface[cn] with QoS class dscp
    else
      send<packet, home-address, flow-id> to cn via M_defInterface[cn] with QoS class dscp-from-upper-layer
  end
end
```

Algorithm 7 The processing in an MN when a user data packet is received

the information in the binding cache. The binding cache is first searched for a binding. If such binding exists, and if it differs from the interface where the packet arrived, a binding update is sent via the care-of address intended for the flow of packets. The packet is then delivered to the upper layers. The variables *x* and *cn* will have the same value in case of route optimization and otherwise *x* will be the HA.

When a packet is sent from upper layers a search is first made for a flow binding. If no such binding exists, the default gateway assigned the *cn* will be used. It can either be the access network used for route optimization or the tunnel to the HA. If the DSCP bits are present in the binding cache they will be used as the QoS class. Otherwise the QoS class expressed by the upper layers is used.

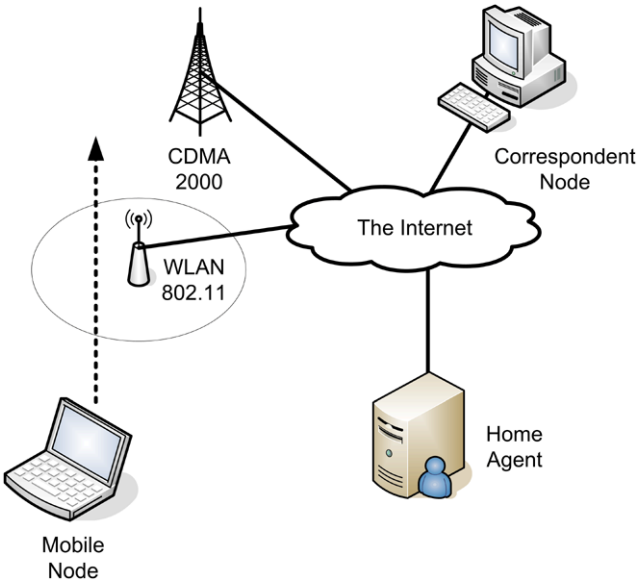


Fig. 19 Evaluation topology

5 Performance evaluations

To evaluate the architecture, a heterogeneous access network topology containing a commercial CDMA2000 network and a WiFi access point was created. The CDMA2000 network operates in the 450 MHz band and the WLAN was an 802.11g access point. Figure 19 illustrates the topology.

Outdoor antennas were used both for WLAN and CDMA2000. The WLAN access point was configured for a capacity of 54 Mb/s giving half the speed in practice. The CDMA2000 network offered bandwidths between 0.5 and 1 Mb/s downlink and 64 kb/s uplink.

The MN was placed in a car traveling at 30 km/h and the start position was out of the WLAN radio range. The car reached the WLAN cell after 25 seconds and after 15 seconds in the coverage of the WLAN connection was lost.

To study handover performance a 120 second VoIP call of 6 kbps two-way traffic was simulated using the Iperf [13], traffic generator.

The experiments conducted used $h = 5$ (in formula (3) and (6)), $c = 5$ (in formula (2)) and hysteresis $= 1$ (the hysteresis constant). The policy cost function in this experiment only used the RNL metric. The BU messages were sent with a one second interval on the CDMA2000 interface and with 100 milliseconds as the interval on the WLAN interface. The selected intervals relate to how fast the MN reacts to variations in RTT and RTT jitter where shorter intervals improve reactivity at the cost of a higher network overhead. A shorter interval is motivated especially for highly fluctuating access networks with high throughput like WLAN.

To evaluate the effectiveness of the architecture a number of parameters were studied. Throughput, delay, delay jitter and packet losses were all studied by looking at the output from Iperf. In Fig. 20, a graph of one experiment shows received bandwidth, jitter for selected access network at each time, and packet loss rate. The calculated policy values for each access network are also plotted as well as information on what interface that was selected at each time. Most notably, packet losses vary from 0 to 4 lost packets for each test in the study. The experiment shown in the graph had 4 packet losses and depicts the worst case performance in the tests. The reason is the lag in calculations and the steep cell edges that come with WLANs. We are solving this by considering a shorter history window in such cases when calculating the policy cost.

RTTs were typically 10 milliseconds for WLAN and 150 milliseconds for CDMA2000 while RTT jitter was typically 10 milliseconds for WLAN and 20 milliseconds for CDMA2000 respectively. With the weights and constants used, the policy value for WLAN varied according to networking conditions with a minimal value around 3.0 while the policy value for CDMA2000 stayed close to 6.0 constantly. Furthermore, the data presented in the graph shows that the bandwidth is very stable during handovers.

To study user-perceived quality parameters the NetAlly software [35] from Viola Networks was used. The G.723.1 codec using approximately 6 kbps was studied for WLAN, CDMA2000, and the combination of those networks. Two-way calls of 60 seconds were studied (see Fig. 21). The first two rows in the figure presents results where the MN is stationary and only CDMA2000 is used. The next two rows are the results from when the MN is in a fixed location in the WLAN network. These two experiments was carried out to benchmark the performance of each access technology. The last two rows present results using our heterogeneous access network approach. In this scenario we can see that the MOS metrics are improved by the heterogeneity compared to only using CDMA2000. With the mobility pattern being out of reach of the WLAN, only using WLAN is not an option.

6 Related work

A class of QoS control frameworks rooted in the IETF is a version of *Policy-based management* (PBM) [34]. PBM provides a way to allocate network resources, primarily network bandwidth, QoS, and security, according to business policies. The European Commission has funded two programs focusing on QoS; *AQUILA* [11, 20] and *TEQUILA* [3]. Both attempt to establish service definition and traffic engineering tools for the Internet to obtain quantitative end-to-end Quality. Tequila tried to establish an IETF working group on formal service level specifications [10]. The Salmon approach is more general in that it provides full modeling capabilities as well as monitoring and calculation

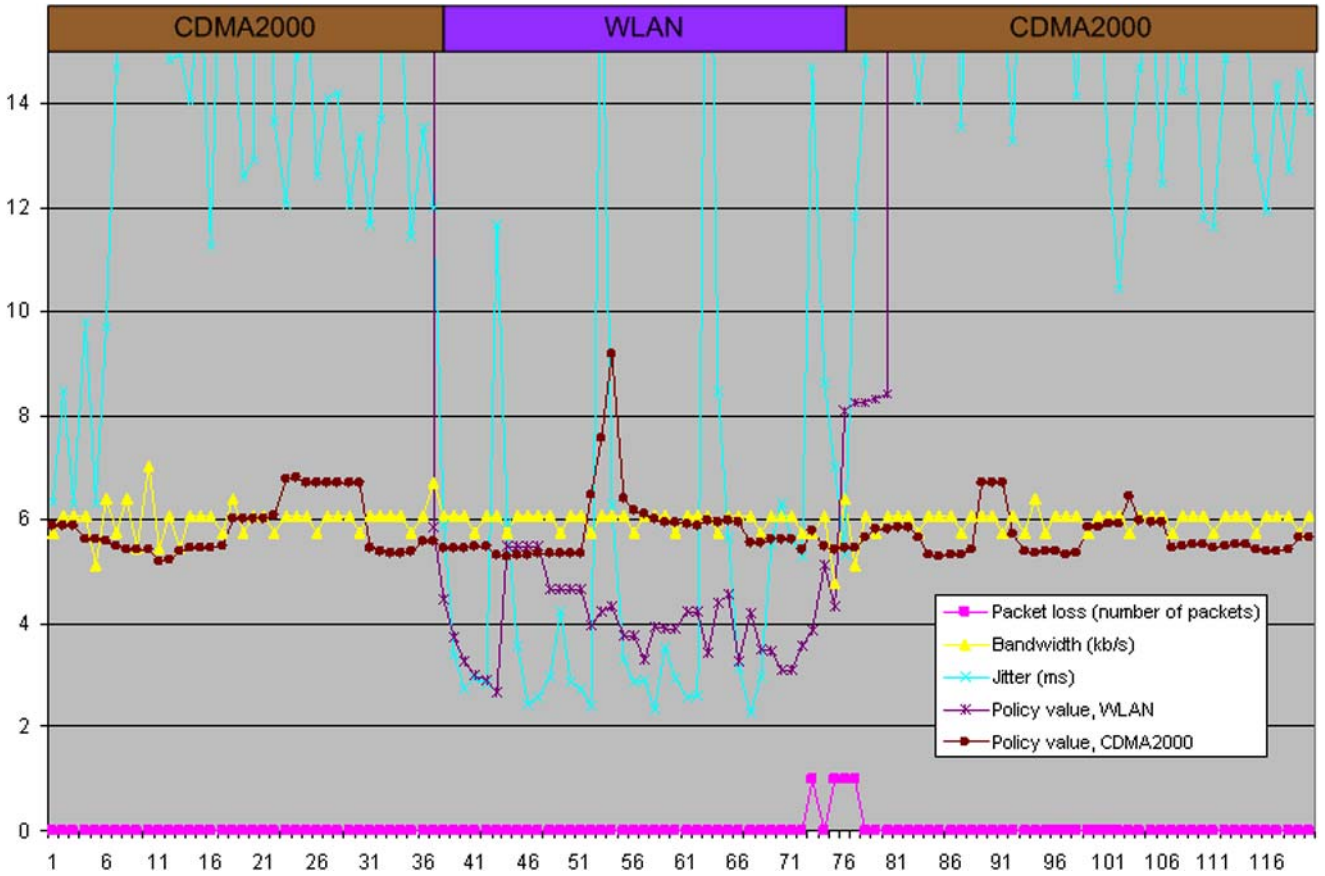


Fig. 20 Results from handover performance studies

	MOS	*R*-value	Jitter (ms)	Delay (ms)	Loss (%)
CDMA2000 (downlink)	3.4	67.2	12.5	104.8	0.03
CDMA2000 (uplink)	2.8	54.9	15.4	169.3	0.02
WLAN (downlink)	3.9	76.1	0.8	9.1	0.02
WLAN (uplink)	3.9	76.1	0.5	9.7	0
Heterogeneous access (downlink)	3.7	72.9	3.8	71.2	0.01
Heterogeneous access (uplink)	3.6	69.6	4.4	70.7	0.01

Fig. 21 Results from study of user-perceived quality of services parameters

capabilities. On the other hand Salmon does not include any control aspects.

Eurescom has funded an end-to-end monitoring effort ;*eQOS* [17], which gives an excellent background description of what we are trying to achieve. *TAPAS* [22] is a similar effort applying SLAs and QoS to application server solutions. It uses SLAng, see below, to define the SLAs. SLAng is focused on applications and is not as general as Salmon.

Probably the most extensive standards effort within service modeling is *CIM* [6, 7]. CIM uses UML as the modeling language, and defines an XML mapping to exchange the models. The key strengths in CIM are the modeling guidelines and patterns used by the standard models and by enterprise extensions. The most important implementation is Microsoft's implementation of CIM in their solution for management of Windows, "*WMI*" [23]. As pointed out by Microsoft in their System Definition Model [24] CIM can "*become unwieldy if used to describe the abstracted virtual constructs of a distributed system*". CIM is aimed more at instrumentation than end-to-end service modeling. IETF is reusing the CIM model in the *IETF Policy Framework WG* [26]. An important part of any service model covering a defined QoS is the SLAs. SLAng [21] is a language focused on

defining formal SLAs in the context of server products like J2EE and typical ASP environments with web services and server applications.

Mitsuya et al. propose a policy management framework for flow distribution in multihomed end nodes [25]. A policy management scenario was defined on top of several multihoming protocols and a number of requirements on such a framework were sorted out. Fan et al. introduce a model for policy-based mobility management based on two coordinated decision engines, one residing in the mobile node and one in the network [9]. Yang et al. propose a QoS-aware routing scheme for heterogeneous wireless networks [38]. Sun et al. present an adaptive connectivity management middleware for heterogeneous wireless networks providing interfaces for applications to query network QoS and availability status, as well as to subscribe to connections events [33].

In Soliman [32] a proposal is presented to lower the delay with MIP messages and thereby manage handover at the network layer more efficiently, considering the time for handovers. The proposal uses two care-of addresses; link local care-of address and regional care-of address. In our solution the possibility of maintaining multiple bindings enables an MN to perform soft handovers.

A solution for fast handovers is presented in Dommety [8]. It uses signalling between the MN, the old AP and the new AP entered to avoid losing packets. Packets will be forwarded from the old AP to the new AP in order to avoid packet losses. Our solution avoids forwarding between APs and complicated signalling between access networks possibly owned by different providers.

In Hseih [12] a combination of the proposals Soliman [32] and Dommety [8] is made, and extended to reduce the handover time even further. The handover time in this work is the same as the handover times for datalink layer handovers. In our case the handover time will be reflected by the responsiveness of detecting link degradation before losing the connectivity.

In Chen [4] a proposal is made for a Smart Decision Model to determine the best available network. The decision model considers factors such as user preferences, system information and properties of available access networks. We are aiming to achieve this with the usage of the Salmon Engine.

Yana [37] proposes an integrated IP-layer handover solution that targets the IP-layer handoff delay. Policies use criteria from user profiles, service requirements and network environment. An adaptive handover control scheme combines probed and monitored (dynamic and static) information. Cross-layer signalling (e.g. L2 triggers) enhances the IP-layer handover. In our work the cross-layer signalling takes place through the Salmon Engine.

A proposal for two flow movement options in MIPv6 is made in Soliman [31]. One is based on IP addresses, protocol and ports and the other is based on the IPv6 flow label. The ideas are similar to ours but the authors do not describe how to manage multiple simultaneous bindings. We also add functionality to prune a flow and to signal DiffServ information (the DSCP bits).

7 Conclusion and future work

To create a competitive market in broadband access there is a need for increased numbers network service providers and a diversity of services offered via service platforms. Customers should be able to access network service providers without pre-subscription and manually entering payment information. Without solutions to overcome the above problems global and seamless connectivity will be hard to achieve. There is also a trend of smaller providers on the market, and roaming solutions without pre-subscription are vital if they are to attract customers. We have presented an architecture that enables this in terms of:

- Defining service levels;
- Managing mobility;
- QoS management and transition;
- Policy management;
- Performance monitoring.

At this stage we assume that users have access to all available access networks. In reality this is not the case. We will in our future work propose an AAA model that enable users to access any access network without pre-subscription.

So far we have only carried out initial experiments and further prototype evaluations are planned. We are working on providing standard Salmon models for important services like VoIP, IPTV broadband access and associated service levels. This goes along with a set of defined modeling guidelines and will create a foundation for a standardized service catalogue. Such a service catalogue needs further investigation into how we can provide a service catalogue across service providers.

From an implementation point of view we are working on the engine interface to provide services for service model operations in contrast with current instance operations. The architecture illustrated in Fig. 5 is implemented using a Windows XP platform.The Salmon Engine enables information exchanges between layers and we intend to develop our ideas even further by exploring and introducing extended signalling between layers in the communication stack.

References

1. Åhlund, C., Brännström, R., & Zaslavsky, A. (2006). Traffic load Metrics for Multihomed Mobile IP and Global Connectivity. *Telecommunication Systems*, *33*(1–3), 155–218.

2. Androutsellis-Theotokis, S., & Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, *36*(4), 335–371.
3. Asgari, A., Trimintzios, P., Irons, M., Pavlou, G., Van den Berghe, S., & Egan, R. (2002). A scalable real-time monitoring system for supporting traffic engineering. In *Proceedings of the IEEE Workshop on IP Operations and Management*.
4. Chen, L. J., Chen, L. J., Sun, T., Chen, B., Rajendran, V., & Gerla, M. (2004). A smart decision model for vertical handoff. In *Proceedings 4th ANWIRE international workshop on wireless Internet and reconfigurability*, Athens, Greece.
5. Deering, S. E. (1991). *ICMP router discovery message*. IETF RFC 1256.
6. DMTF (2007a). *Directory enabled networks*. http://www.dmtf.org/standards/wbem/den.
7. DMTF (2007b). *Common information model 2.17, 2007*. http://www.dmtf.org/standards/cim/.
8. Dommety, G. et al. (2004). *Fast Handover for Mobile IPv6*. Internet draft.
9. Fan, C., Schlager, M., Udugama, A., Pangboonyanon, V., Toker, A. C., & Coskun, G. (2007). Managing heterogeneous access networks coordinated policy based decision engines for mobility management. In *Proceedings of the 32nd IEEE Conference on Local Computer Networks (LCN 2007)* (pp. 651–660).
10. Goderis, D., T'joens, Y., Jacquenet, C., Memenios, G., Pavlou, G., Egan, R., Griffin, D., Georgatsos, P., Georgiadis, L., & Heuven, P. V. (2000). *Service level specification semantics and parameters*. http://www.ist-tequila.org/standards/draft-tequila-sls-00.txt.
11. Granzer, H. (2003). *Aquila: adaptive resource control for QoS using an IP-based layered architecture*.
12. Hseih, R., Zhou, Z. G., & Seneviratn, A. (2003). S-MIP: a seamless handoff architecture for mobile IP. In *IEEE INFOCOM 2003 – The conference on computer communications* (pp. 1774–1784).
13. Iperf (2007). *Version 1.7.0 and version 2.0.2*. dast.nlanr.net/Projects/Iperf.
14. ITU-T (1996). *Recommendation P. 800*. Methods for subjective determination of transmission quality.
15. ITU-T (1999). *Recommendation G. 109*. Definitions of categories of speech transmission quality.
16. ITU-T (2003). *Recommendation G. 107*. The E-model, a computational model for use in transmission planning.
17. Jensen, T. (1999). Managing quality of service in multi-provider environment. In *Proceedings of Telecom 99*.
18. Johnson, D. B., Perkins, C. E., & Arkko, J. (2004). *Mobility support in IPv6*. IETF RFC3775.
19. Kent, S., & Seo, K. (2004). *Security architecture for the Internet protocol*. Internet Draft, Dec, 2004.
20. Koch, B. F., & Hussman, H. (2003). Overview of the project AQUILA (IST-1999-10077). In *Proceedings of architectures for quality of service in the Internet*.
21. Lamanna, D. D., Skene, J., & Emmerich, W. (2003). SLAng: a language for service level agreements. In *Proceedings of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems*.
22. Lodi, G., Panzieri, F., Rossi, D., & Turrini, E. (2005). Experimental evaluation of a QoS-aware application server. In *Proceeding of the 4th IEEE International Symposium on Network Computing and Applications*.
23. Microsoft (2006a). *WMI – Windows Managment Instrumentation*. http://www.microsoft.com/whdc/system/pnppwr/wmi/default.mspx.
24. Microsoft (2006b). *System definition model*. http://www.microsoft.com/windowsserversystem/dsi/sdm.mspx.
25. Mitsuya, K., Kuntz, R., Sugimoto, S., Wakikawa, R., & Murai, J. (2007). A policy management framework for flow distribution on multihomed end nodes. In *Proceedings of the 2nd ACM International Workshop on Mobility in the Evolving Internet Architecture*.
26. Moore, B., Ellesson, E., & Strassner, J. (2001). *Policy core information model – version 1 specification*. IETF RFC3060.
27. Moore, B., Ellesson, E., Strassner, J., & Westerinen, A. (2001). *Policy core information model – version 1 specification*. IETF RFC 3060.
28. Nasser, N., Hasswa, A., & Hassanein, H. (2006). Handoffs in fourth generation heterogeneous networks. *IEEE Communications Magazine*, *44*(10), 96–103.
29. Perkins, C. E. (2002). Mobile IP. *IEEE Communications Magazine*, *40*(5), 66–82.
30. Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (2003). *RTP: a transport protocol for real-time applications*. IETF, RFC 3550.
31. Soliman, H., Malki, K. E., & Castelluccia, C. (2003). *Flow movement in Mobile IPv6*. Internet draft.
32. Soliman, H., Catelluccia, C., Malki, K. E., & Bellier, L. (2004). *Hierarchical MIPv6 mobility management*. Internet draft.
33. Sun, J.-Z., Riekki, J., Jurmu, M., & Sauvola, J. (2005). Adaptive connectivity management middleware for heterogeneous wireless networks. *IEEE Wireless Communications*, *12*(6), 18–25.
34. Verma, D. C. (2002). *Simplifying network administration using policy-based management*. Network, IEEE, pp. 20–26.
35. Viola (2007). *NetAlly Base System, SMB Edition, version 5.2.31*. www.violanetworks.com.
36. Wallin, S., & Leijon, V. (2007). Multi-purpose models for QoS monitoring. In *Proceedings of 21st international conference on advanced information networking and applications workshop: AINAW'07* (Vol. 1, pp. 900–905). IEEE Computer Society.
37. Yana, B., Jianwen, H., Iver, P., Mei, S., & Song, J. (2004). An integrated IP-layer handover solution for next generation IP-based wireless network. In *Proceedings of the 60th IEEE vehicular technology conference*.
38. Yang, K., Wu, Y., & Chen, H. H. (2007). QoS-aware routing in emerging heterogeneous wireless networks. *IEEE Communications Magazine*, *45*(2), 74–80.
39. Yavatkar, R., Pendarakis, D., & Guerin, R. (2000). *A framework for policy-based admission control*. IETF RFC 2753.

Christer Åhlund received his M.Sc. in Computer Science from Uppsala University and his Ph.D. from Luleå University of Technology (LTU). Christer is currently head of division for Mobile Networking and Computing at LTU Skellefteå. His research is about IP mobility, ad-hoc networks, heterogeneous access networks and media distribution. He has been working in the Industry for 10 years as a programmer, system engineer and project manager. Christer is also working part time in the telecommunication industry.

Stefan Wallin is a network management solution architect and senior partner at Data Ductus and a researcher at Luleå University of Technology. He received his M.Sc. from Linkoping University at 1989. Since then he has worked with network management, initially at Ericsson and the last 12 years at Data Ductus. The research focus is service modeling and monitoring in parallell with alarm analysis in mobile networks. The research work is funded by Data Ductus.

Karl Andersson received his M.Sc. in computer science and computer engineering from Royal Institute of Technology, Stockholm, Sweden, in 1993. After spending more than 10 years working as a system developer, system designer, project manager and business manager within the Capgemini group working mainly with telecom clients Karl returned to academia in 2005. Karl is currently a Ph.D. student in media technology at the division of Mobile Networking and Computing, Luleå University of Technology, specializing in multimedia distribution in heterogeneous networks.

Robert Brännström received his B.Sc. in Computer Engineering and his Ph.D. in Computer Science from Luleå University of Technology (LTU), in 2001 and 2007. The Ph.D. is performed within the research subject of Media Technology and is focused on mobility management in heterogeneous access networks. Robert is holding the position as lecturer and second head of the division for Mobile Networking and Computing at LTU Skellefteå. At the University he is working with research and lecturing within computer communication and software engineering. Robert also holds a certification to teach within the Cisco academy program, Cisco Certified Network Associate (CCNA). He is a member of IEEE Computer and Communication Society. His research interests include IP networks, Mobility management and heterogeneous wireless communications. He has been session chair and member of the technical programme committee at several conferences. The research has been promoted by EU funding, Swedish research funding and local companies.