Autonomic System for Mobility Support in 4G Networks

Pablo Vidales, *Student Member, IEEE* , Javier Baliosian, Joan Serrat, *Member, IEEE,*
Glenford Mapp, Frank Stajano, Andy Hopper, *Fellow, IEEE*

***Abstract*— Individual wireless access networks show limitations that can be overcome through the integration of different technologies into a unified platform (i.e. 4G system). Nevertheless, the integration of heterogeneous networks poses many challenges such as adding complexity to the processes of deciding when to handoff, selecting the best network, and minimising roaming effects using appropriate handover methods. This paper presents PROTON, a novel solution that assists mobile users in the decision-making process related to roaming between heterogeneous technologies. PROTON deploys a formal policy representation model, based on Finite State Transducers, that evaluates policies using information from the context to manage mobiles' behaviour in a transparent manner, hiding 4G systems' complexities. We blend concepts of autonomic computing into the design of the solution and manage to improve user experience in typical 4G scenarios while keeping transparency.**

***Index Terms*— policy systems, 4G networks, Finite State Transducer, heterogeneous, handover.**

I. Introduction

THE ubiquitous explosion of Internet services and the rapid proliferation of mobile networked devices, as well as radio access technologies, creates a unique challenge for networking researchers. The next generation of communication systems will involve mobile users interacting with a pervasive computing environment that adapts accordingly. New solutions are required for managing interactions among the plethora of inter-connected networks, wireless devices and IP-based services.

There is a wide range of wireless access networks becoming available such as infrared, bluetooth, 802.11-based wireless LANs, cellular wireless, and satellite networks, which will combine to provide a highly integrated wireless access platform. Katz, et al., termed this model as Wireless Overlay Networks [1]. The wireless networks that form the overlay have different characteristics, and there is a trade-off associated between bandwidth and coverage (typically, smaller/local coverage has higher bandwidth).

The evolution in wireless access technologies shows that the trade-offs between coverage and bandwidth will exist. Ideally, a wireless access technology with unlimited coverage and infinite bandwidth would be desirable. Since this is not easy to achieve (due to spectrum and mobility constraints), researchers are focusing on creating an integrated platform architecture able to emulate the *perfect* wireless access network for mobile users. Thus, the vision for the next generation of wireless architecture (4G) builds on the key notion of *heterogeneous wireless integration and inter-networking.*

Pablo Vidales, Dr. Glenford Mapp, Dr. Frank Stajano and Prof. Andy Hopper are members of the Laboratory for Communication Engineering at the University of Cambridge (United Kingdom)

Javier Baliosian and Joan Serrat are with the Network Management Group at Polytechnic University of Catalonia (Spain)

TABLE I
DIVERSITY IN EXISTING AND EMERGING WIRELESS TECHNOLOGIES DEMAND FLEXIBLE AND ADAPTIVE ROAMING DEVICES.

Network	Coverage	Data Rates	Cost
Satellite (B-GAN)	World	Max. 144 kb/s	High
GSM/GPRS	Aprox. 35 Km	9.6 kb/s up to 144 kb/s	High
IEEE 802.16a	Aprox. 30 Km	Max. 70 Mb/s	Medium
IEEE 802.20	Aprox. 20 Km	1-9 Mb/s	High
UMTS	20 Km	up to 2 Mb/s	High
IEEE 802.11g	100 - 300 m	54 Mb/s	Low
HIPERLAN 2	70 up to 300 m	25 Mb/s	Low
IEEE 802.11a	50 up to 300 m	54 Mb/s	Low
IEEE 802.11b	50 up to 300 m	11 Mb/s	Low
Bluetooth	10 m	Max. 700 kb/s	Low

Due to the multiplicity of choices available from many cellular/wireless network providers, access technologies, mobile devices, and disparate services requirements, there is a significant need to address all of this as a single integration challenge. The 4G architecture envisions highly flexible and adaptive integration of diverse mobile client systems and network technologies to support built-in capability for seamless interaction in this pervasive computing environment.

Implicitly, this also means that there will be a need for mobile devices that can cope with the complexity and dynamics of the next generation (4G) of wireless access environments. With more technologies, services, and devices joining the fray, we can expect that the gap between the service levels offered by new access networks will close, adding more complexity to the networking process (see table I). We consider that the system-embedded handover policy "always switch to the smallest-coverage overlay" becomes invalid as QoS gaps narrow.

Also, the growth in the popularity of Internet services among mobile users, together with the higher QoS required by novel applications, demands improving resource management capabilities in mobile devices to offer a better user experience. High mobility, seamless roaming, high data access rates and transparent connectivity to services from "any" device are dominant trends in the 4G vision and the basic reasons to think that *autonomic computing* means a plausible solution for emerging challenges.

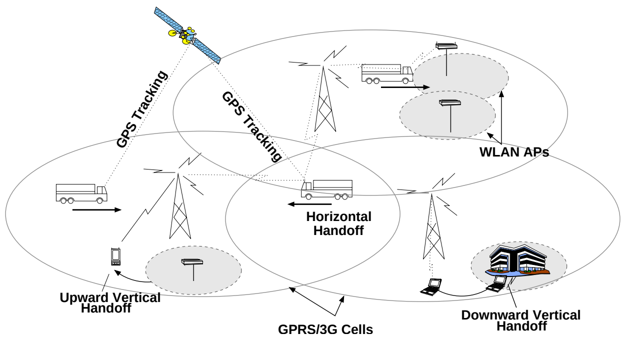


Fig. 1. Future 4G communication system.

In this sense, autonomic computing is an approach to self-managed systems with a minimum of human interference. This new computing paradigm means that the design and implementation of an autonomic system must exhibit these fundamentals from the user perspective: flexibility, accessibility, and transparency [2].

We hold that some principles of autonomic computing should be applied to the design of solutions to support mobility and deal with complexity in the next generation of wireless networks. This paper describes PROTON [1] [3], a solution that blends concepts of autonomic computing, policy-based systems, and a novel model based on Finite State Automata (FSA), to solve mobility management issues in 4G networks.

This FSA-based model uses a new metric called *Tautness Function* (TF), and a new kind of automata called *Finite State Transducer with Tautness Functions and Identities* (TFFST) [4]. The TF and the TFFST were defined to model policies and resolve potential conflicts. Conflict resolution has been one of the main obstacles for policy-based systems and our model handles conflicts with good run-time performance while greatly reducing human intervention.

A. The Problem: Seamless Complexity

Future wireless environments will not consist simply of one radio access technology such as current cellular systems (e.g., GSM, WCDMA, or EDGE), but will integrate multiple access networks, adding complexity to mobility management systems. Moreover, seamless inter-networking (as shown in Figure 1) will be a basic feature in mobile terminals to allow connectivity in this pervasive computing environment.

Giving such capability to users across heterogeneous networks is much more complicated than in homogeneous scenarios. In this case, where multiple disparate networks are accessible from a mobile terminal, detecting the possible options and choosing the optimal combination of network resources and active applications at the correct moment, becomes a complex procedure.

In contrast to traditional algorithms, mobility management systems will need many parameters to support vertical handover-related processes. Table II shows the main challenges in 4G systems, mobile devices need more intelligent solutions to handle these complexities, while maintaining transparency to avoid affecting usability.

[1]*P*olicy-based system to *RO*am *T*ransparently among *O*verlay *N*etworks

TABLE II

THERE ARE COMPLEXITIES THAT STEM FROM 4G SYSTEMS, COMPARED WITH CURRENT HOMOGENEOUS ENVIRONMENTS.

Homogeneous Networks	**Heterogeneous Networks**
Detection of available access points in the current system.	Detection of access points in the available networks.
Mobile host needs to decide among access points of the same technology.	Mobile host needs to decide among access points of multiple technologies.
Handover initiation triggered mainly by signal strength fading.	Handover initiation triggered by multiple events.
The execution methods can be applied in every situation.	The execution methods depend on context and not all methods can be applied in every scenario.
Adaptation process is not as important because the mobile host roams between similar conditions (same technology).	Adaptation is essential, the mobile host roams between disparate technologies and conditions change drastically.

B. Autonomic Solution for 4G Systems

IBM research outlined eight defining characteristics of an autonomic system. We sense that taking into account heterogeneity, dynamics, and complexity added in 4G environments, an appropriate support should endeavour to possess these key elements with the intention of offering a complete seamless solution [2]. From these concepts, we integrate the following characteristics in PROTON's design:

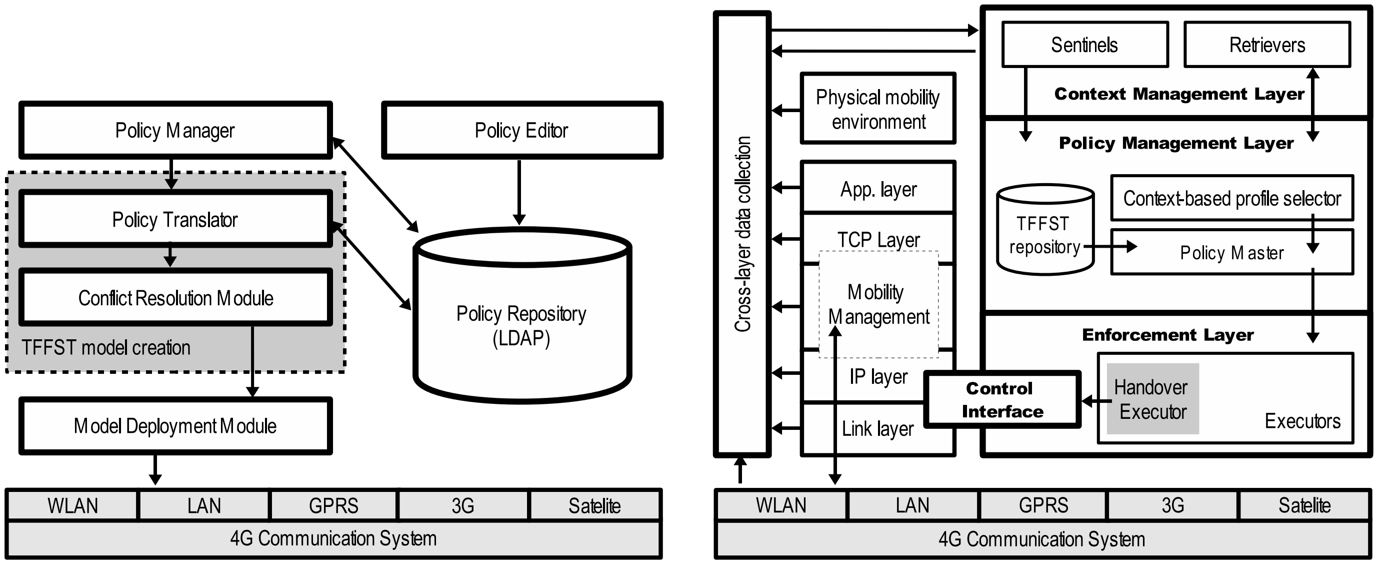
To be autonomic, a system needs to "know itself". An autonomic system will need detailed knowledge of its components, current status, and ultimate capacity, as well as possible connections with other systems. PROTON's architecture (described in Section II) allows the system to access a detailed *Networking Context*, which includes important data about mobile host's network resources, activity, physical environment, as well as users' preferences at all times. This gives the device capability to know the extent of its own resources and decide how to use them.

An autonomic system must configure and reconfigure itself under varying and unpredictable conditions. PROTON uses the knowledge about its context (i.e. Networking Context) to feed a policy-based model that controls terminals' initial configuration as well as its ongoing behaviour according to the generated events (e.g., connection/disconnection, activity variations, and users' preferences changes).

An autonomic system never settles for the status quo – it always looks for ways to optimise its workings. In this sense, considering dynamics in the conditions when dealing with mobility, PROTON always senses the environment and evaluates policies to look for the best possible relation between terminal activity and connectivity resources.

An autonomic system knows its environment and context surrounding its activity, and acts accordingly. It is essential for PROTON to sense its context and produce events to trigger policies that drive a mobile's behaviour.

An autonomic system cannot exist in a hermetic environment. In this sense, PROTON is compatible with the TCP/IP stack and it helps in the integration process of heterogeneous networks, creating an open IP-based platform to access mobile services.



(a) Network-side components (b) Host-side components

Fig. 2. PROTON's architecture stems from the vision of 4G communication networks. The three-layer system, together with an adequate context gathering and policy deployment model, copes with dynamics and complexity in future integrated heterogeneous networks.

An autonomic system will anticipate the optimised resources needed while keeping its complexity hidden. PROTON offers seamless mobility support, coping with the complexity posed by 4G systems, hiding it from the users.

Currently, a system incorporating the eight elements [2] will be very difficult to build; however, we do believe that the solution presented in this paper can be considered as an early attempt to critically examine such concepts. An autonomic system seems appropriate to tackle the complexity posed by future integrated heterogeneous environments formed by diverse access networks and services, and a huge variety of mobile terminals interacting.

The next section describes PROTON's architecture that is divided into network- and host-side components. Section III introduces the concept of Networking Context, defining the three datasets that form it. Then, Section IV explains the policy model based on Finite State Transducers. In Section V, we describe the processes related to the generation, deployment, and evaluation of TFFSTs. Section VI details the policy enforcement layer, describing how actions are executed on the LCE-CL testbed. We present the evaluation results in Section VII and related work in Section VIII. Finally, future research is mentioned in Section IX, and we conclude in Section X.

II. Architecture

PROTON components are divided into network-side and host-side components. The reason for this is that because of the number of decisions required to fully support the handover process, the raw policy set can get too complex to maintain within a limited mobile device. However, the functionality still being completely based on the mobile host, only the highly demanding pre-processing tasks related to the policy evaluation model are placed in the network – where computing constraints are much more relaxed (see Figure 2(a)).

The host-side components are organised into a three-layered system: *Context Management layer*, *Policy Management layer*, and *Enforcement layer*, which sit on top of Layer 3 in the protocol stack. The network-side contains the components related to the specification and deployment of the policies.

A. Network-side components

Those components that involve operator's management or high computational cost are located in the network to minimise complexities at the mobile terminal. This is the case of policy definition, storage, and conflict resolution. Network-side components are shown in Figure 2(a).

Policy Editor – To create the system policies, the operator must write them in a high level policy specification language. We chose Ponder as the high-level language because of its expressiveness and deployed tools. In particular, in PROTON we have used the *Ponder Policy Editor* and its compiler [5] to create the first internal Java representation of the policies.

Policy Repository – The policy repository is implemented using a Light-weight Directory Access Protocol (LDAP) server, which intends to store system policies in their high-level representation as well as in the internal Java representation.

Policy Translator – This component translates the policies specified in Ponder language [6] into the evaluation model described in Subsection IV-C.

Conflict Resolution Module (CRM) – The conflict resolution module builds the deterministic Finite State Machine modelling every active policy. The CRM performs two main tasks: (1) it combines the policies among them considering the system constraints and (2) it resolves conflicts among those rules. During this task, all possible static and dynamic conflicts are foreseen. Therefore, the algorithms that are executed have a high computational cost. The main benefit of adding such overhead on the network-side is to avoid heavy tasks in the mobile device (usually a terminal with limited computational power and memory capacity). Furthermore, after resolving conflicts and constructing the deterministic Finite State Machine, the mobile device can react quickly to incoming events.

Model Deployment Module (MDM) – Once all policies and constraints are combined in a TFFST, it is delivered to the mobile device and installed into its *policy master* to drive its decisions. One TFFST is created and deployed for each mobility profile and this module takes care of coding and transmitting the transducers to each mobile device. This process is carried out sending the Java object via RMI.

B. Host-side components

Context Management Layer (CML) – Figure 2 shows PROTON architecture, the top layer has two type of components to obtain the Networking Context: *Sentinels* and *Retrievers*. The former is responsible for collecting dynamic elements, and the latter manages static elements. There is a responsible object for each context element, and it has individual settings (e.g., polling frequency and local rules) depending on the complexity and dynamics of a particular fragment. For example, *VelocitySentinel* polls the velocity every second due to the constraints in the GPS receiver. The local rule (shown in Section III) filters the collected data according to the current velocity and acceleration. Thus, not every reported measurement generates an event.

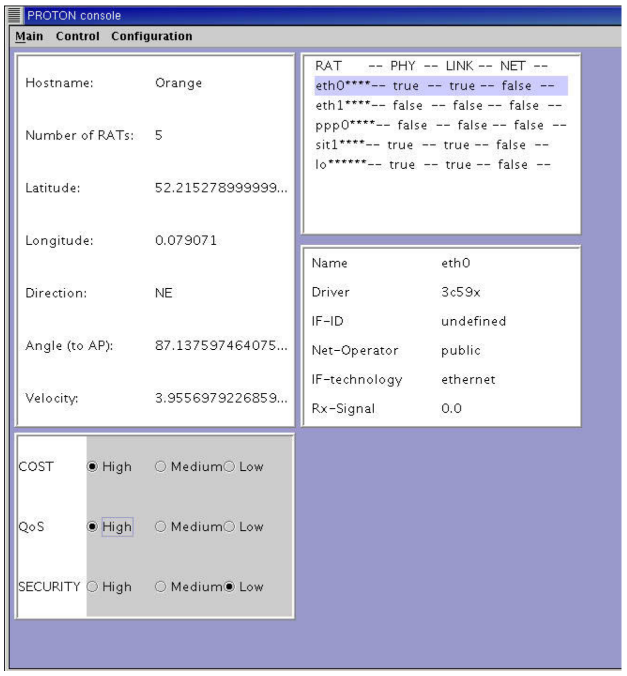
Policy Management Layer (PML) – Responsible for the control and evaluation of the policies to drive the behaviour of the mobile device. It is composed of the following elements:

Policy Master: This component acts as the Policy Decision Point (PDP) in the policy system [7]. It receives events (e.g., Transition-Pedestrian produce by the VelocitySentinel) from the CML, and according to these inputs, it decides the possible actions to execute, which are immediately sent to the Enforcement Layer.

Context-based profile selector: The fact that only a small portion of sensory input is relevant under certain conditions is used to improve the performance of the system. Some inputs can generate special events (i.e. *macro-events*) which are then used by the selector to load a profile that defines a valid subset of policies to evaluate, i.e. the appropriate TFFST . An example of a macro-event is velocity – if host speed is more than 90km/h the only active policies are those that produce an upward handover as an action. This means that mobile users should never attempt to connect to a lower layer when moving at very high speeds.

TFFST Repository: The TFFSTs are produced in the network side, as mentioned in Subsection II-A, and then deployed into the mobile device where they are kept in the TFFST repository. Thereafter, the selected TFFST and its evaluation are decided according to the events received from the CML.

Enforcement Layer (EL) – Formed by different *Executors* that are the Policy Enforcement Points (PEPs) of the system



Hostname:	Orange
Number of RATs:	5
Latitude:	52.215278999999...
Longitude:	0.079071
Direction:	NE
Angle (to AP):	87.137597464075...
Velocity:	3.9556979226859...

```
RAT     -- PHY  -- LINK -- NET --
eth0****-- true  -- true -- false --
eth1****-- false -- false -- false --
ppp0****-- false -- false -- false --
sit1****-- true  -- true -- false --
lo******-- true  -- true -- false --
```

Name	eth0
Driver	3c59x
IF-ID	undefined
Net-Operator	public
IF-technology	ethernet
Rx-Signal	0.0

COST [✓ High] Medium Low

QoS [✓ High] Medium Low

SECURITY High Medium [✓ Low]

Fig. 3. PROTON Networking Context monitoring console.

[7]. They are responsible for performing the actions that result from evaluating the TFFST. The EL connects with the lower layers through a *Control Interface* (CI) that captures incoming router advertisements just before they reach the Mobile IPv6 module –prior to the handover procedure. The CI executes different scripts, which receive the selected interface as a parameter and outline the execution handover method.

Communication protocols – For the connection CML-PML and the communication within the PML, we use a generic asynchronous notification service called *Elvin* [8]. This service was primary designed as a middleware for distributed systems, however, many research projects have used Elvin due to its simplicity. Ponder uses this messaging service in its framework, and we decided to use it in our system as well.

III. Networking Context

Context is defined as any information sensed from the environment which may be used to define the behaviour of a system. The effectiveness of PROTON's assistance depends on three main tasks: accurate extraction, combination, and expression of unsteady measurements collected from the environment. These tasks are constrained by three factors: frequency in sensory capture, complexity in context fusion, and limited inference capability, respectively.

Since in a highly dynamic environment, the instability of the sensed data has a negative impact on the amount of information that can be extracted from a particular context fragment, PROTON organises sensed data (i.e. Networking Context) into a three-level hierarchy according to: *dynamics of sensed data* and *complexity of the rules applied*. This taxonomy results in the definition of three datasets, each of which has a particular combination of rules' complexity and components' dynamics.

Collected dataset – Every dynamic fragment gathered by a sentinel is part of this dataset (high and medium dynamics components). Sentinels poll data from many different sources, and then filter it according to simple local rules that only affect the specific context element. The output of the collected dataset is smaller than the input, which reduces the processing overhead in the mobile device. For example, the local rule shown below corresponds to the VelocitySentinel, it filters the collected data (every second) according to current speed and the increment in velocity. Therefore, this minimises processing and assures that generated events respond to meaningful context changes.

```
voidlocalRule(double currentVelocity) {
  double velDiff;
  velDiff = Math.abs(currentVelocity - Host.getVelocity());
  /*IF Velocity is lower than PEDESTRIAN AND change in velocity is higher
  than 2.5km/hr*/
  if (currentVelocity < PEDESTRIAN & velDiff > 2.5)
    /*Generate event Transition-pedestrian*/
    event = new NamedEvent();
    String[] params = new String[10];
    params[0] = "Transition-Pedestrian";
    params[1] = "double";
    params[2] = Double.toString(velDiff);
    event.HandleEvent(params);
  if (currentVelocity < LOW_AUTOMOBILE & velDiff > 5)
    /*Generate event Transition-low-automobile*/
  if (currentVelocity < HIGH_AUTOMOBILE & velDiff > 10)
    /*Generate event Transition-high-automobile*/
  if (currentVelocity > HIGH_AUTOMOBILE & velDiff > 20)
    /*Generate event Transition-high-speed*/
}
```

Aggregated dataset – It groups the filtered and retrieved information coming from the CML. The former is the output of the collected dataset after applying the corresponding local rules. The latter derives from the low-dynamic components, which are managed by Retrievers, e.g., user preferences retriever or application profile retriever.

Networking Context dataset – It is a snapshot of the Aggregated and Collected datasets used by the Policy Master to select the path and evaluate the conditions in the TFFST. The Networking Context (see Figure 3) allows the mobile host to have complete knowledge of its resources, context, and activity at all times.

IV. Policy Model

A. Motivating the use of Policies

Multimode mobile devices must be flexible and proactive to cope with dynamics and changes in 4G systems. PROTON has to cover several aspects that derive from this premise:

- *The solution must include physical context (e.g., velocity and position),*
- *Adaptation must be supported in the system,*
- *PROTON must lead to unambiguous decisions in the shortest possible time,*

After pondering these requirements, we decided that an effective approach to address the problem is a policy-based system to assist users in future mobile scenarios. Moreover, considering the constraints of dynamics and complexity, we broke context into simpler and more intuitive fragments (as shown in Section III) and wrote policies using these elements as conditions.

Thus, complexity is transferred to the combination of policies and decision-making, instead of having it in the individual rules. Therefore, using a policy-based system enables easier tuning of the system's behaviour. Employing cost functions to drive decisions can often lead to static and over-complicated solutions, as the complexity is related to the number of parameters.

Furthermore, breaking down context into fragments allows us to use independent normalisation functions for each element. This leads to a more accurate transformation of the parameters, while cost functions are more static. In conclusion, a policy-based system is more flexible and can express more than a cost function.

Our policy model uses Ponder [6] as a high-level language for policy specification. This framework is used to obtain an initial Java representation from the high-level policy. The Ponder language provides a common means of specifying policies that map onto various actors within a network. However, adaptations are required in order to use Ponder in a particular application as the implementation of an autonomic solution for 4G systems.

B. Policy Specification

PROTON follows the *Event-Condition-Action* (ECA) paradigm where policies are rules that specify actions to be performed in response to predefined conditions, triggered by events (see sample policy below).

```
  Rule 1:
inst oblig /ProtonPolicies/Obligs/CheckupPolicy {
  on PhysicalConnection(nic);
  subject /ProtonPMAs/HandoverPMA;
  target t = /ProtonTargets/HandoverExecutor;
  do t.networkSelectionEvent(nic);
  when t.isLinked(nic);
}
```

The policy shown above, *CheckupPolicy*, is triggered when a new radio access interface is connected to the mobile host – the event *PhysicalConnection* is sent by the *AttachedSentinel*. The policy target, *HandoverExecutor*, checks the connectivity in the Network Interface Card (NIC) executing the method *isLinked(nic)*. Then, if the new NIC is ready to transmit and can be considered as an option, the policy target sends an event to initiate the process of network selection by executing the method *networkSelectionEvent(nic)*. This high-level policy is compiled into an initial Java representation and translated into TFFSTs.

C. An Evaluation Model Based on Finite State Transducers

Finite State Automata are classical computational devices used in a variety of large-scale applications. FSTs, in particular, are automata whose transitions are labelled with both an input and an output label. They have been useful in a wide range of fields, but particularly in Natural Language Processing. This discipline makes intensive use of grammatical rules, which are ambiguous by nature, and requires quick decisions based on those rules, in particular in fields such as speech recognition with major performance requirements.

Additionally, Finite State Machine-based solutions are typically light-weight. They can be implemented as arrays of states, transitions, and pointers among them without falling into heavy management structures.

We represent the policies as *deterministic transducers* that are a category of transducers without ambiguities. This means that at any state of such transducers, only one outgoing arc has a label with a given symbol or class of symbols.

Deterministic transducers are computationally interesting because their computation order does not depend on the size of the transducer, but rather only on the length of the input since the computation consists of following the only possible path corresponding to the input and writing consecutive output labels along the path [9].

For representing policies with FSTs we used the model presented in [4]. It is based on a modification of predicate augmented FSTs [10], in which predicates were replaced by a metric representing a sort of distance between a policy and a given event.

A policy has a condition delimiting a region where a given event can or cannot lie. When such an event is inside two or more overlapping regions a modality conflict may arise. We are concerned about how tautly a condition fits to an event instead of how far from the border it is. Thus, our preferred condition will be that which is the most *taut* around the event under consideration.

In order to quantitatively represent the aforementioned *tautness*, we use the metric called Tautness Function, a real number in the interval $[-1, 1]$ so that the more taut a condition is, the closer its TF is to zero.

Definition 1 *A Tautness Function associated with a condition c, denoted τ_c, establishes a mapping from $E \times C$ to the real interval $[-1, 1]$ where:*

- *E is the set of possible network events or attempted actions,*
- *C is the set of policy conditions,*
- $\tau_c(e) \in [-1, 0) \Leftrightarrow e\, not\, satisfies\, c,$
- $\tau_c(e) \in (0, 1] \Leftrightarrow e\, satisfies\, c,$
- $\tau_c(e) = 1 \Leftrightarrow \forall f \in E, f\, satisfies\, c,$

When the TF is modelling the condition part of the rule, we include in condition c the subject or any other property of the condition such as temporal constraints. In the same manner, when the TF is modelling the action part of the rule, condition c includes the target or any property of the action

To provide an intuitive example of TF, let us assume that one policy specifies *wireless interfaces* in general and another policy specifies *IEEE 802.11b interface* (a subset of *wireless interfaces*). For an action attempted by a IEEE 802.11b interface, the second policy should define a TF that is closer to 0 than the first policy. However, as with the distance-to-a-policy concept, much more complicated expressions could be computed, for example using the associated traffic types to the interface or the QoS characteristics.

Notice that in the TF definition we are stating only the general rules with which a TF should comply. This non-specificity is deliberate, because how it must be implemented or how it maps events and conditions to real numbers should be decided in the context of a specific policy-based system and technology. Thus, a TF is an abstraction layer of technology-dependent issues that allow us to work in a more general fashion.

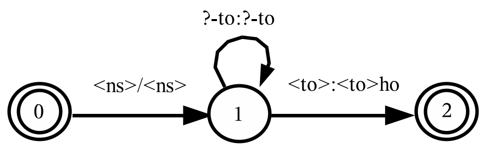


Fig. 4. TFFST model for the obligation in Rule 1 (**ns** is *NetworkSelected*, **to** is *TimerOver*, and **ho** is *handoff*)

In Subsection V-F we show some examples of how we compute TFs. The most outstanding advantage of using TFs in PROTON is the capacity to define a different way to ponder each networking context fragment and combine them using the algebra for TFs, which defines the basic logic operators *disjunction*, *conjunction* and *negation*, plus two new operators called *tauter-than* ($\rightarrow_\tau$) and *as-taut-as* ($\rightleftarrows_\tau$) specially formulated to express the concept of distance in the TFs (see detailed algebra definition in [4]). Below we define the transducers that use TFs to model policies internally on the host side.

Definition 2 *A Finite State Transducer with tautness functions and identities (TFFST) M is a tuple (Q, E, T, Π, S, F) where:*

- *Q is a finite set of states,*
- *E is a set of symbols,*
- *T is a set of tautness functions over E.*
- *Π is a finite set of transitions $Q \times (T \cup \{\epsilon\}) \times (T \cup \{\epsilon\}) \times Q \times \{-1, 0, 1\}^2$.*
- *$S \subseteq Q$ is a set of start states.*
- *$F \subseteq Q$ is a set of final states.*
- *For all transitions $(p, d, r, q, 1)$ it must be the case that $d = r \neq \epsilon$.*

In the implementation, we use an extension of the above definition to let the transducer deal with strings of events and actions in each transition. Policy rules are modelled using TFFSTs, in which the incoming label represents the condition and the outgoing label the action.

D. Modelling Policies with TFFSTs

To understand how the entities introduced before are used for modelling policies, we present how *obligations* and *constraints* are expressed. TFFSTs may model authorisations, prohibitions and dispensations as well, but the following two policy types are expressive enough to deal with current PROTON requirements.

Obligations – An *obligation* is a rule expressing that when an event fulfils a particular condition, a given action must be executed. It is represented as a transducer with a main link that has an event as the input and the action as the output.

[2]The final component of a transition is an "identity flag" used to indicate when an incoming event must be replicated in the output.

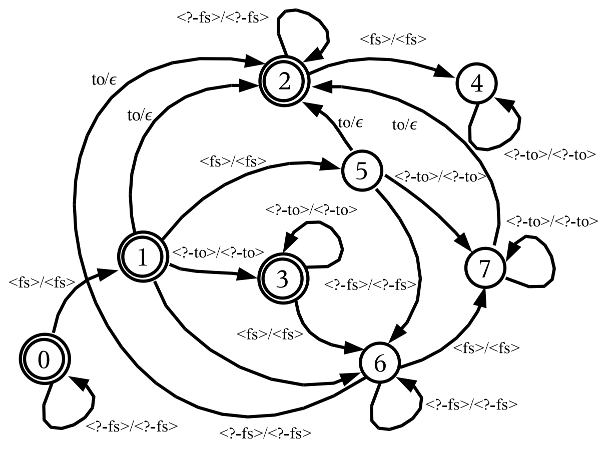


Fig. 5. TFFST model for the constraint in Rule 2 (**ns** is *NetworkSelected*, **to** is *TimerOver*, **ho** is *handoff*, and **fs** is *FadingSignal*)

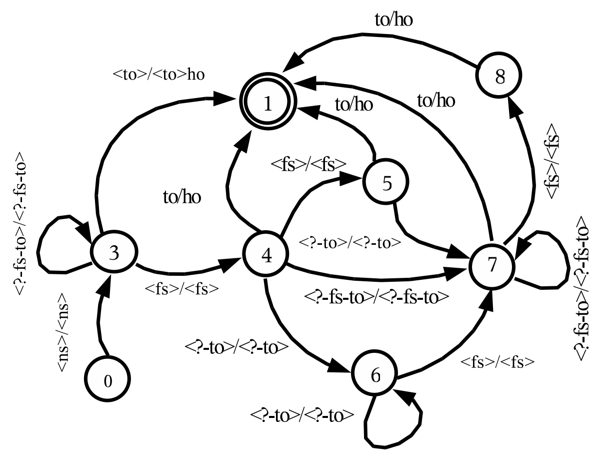


Fig. 6. TFFST model for composition of rules 1 and 2 (**ns** is *NetworkSelected*, **to** is *TimerOver*, **ho** is *handoff*, and **fs** is *FadingSignal*)

Typically, the incoming event will report the occurrence of a fact and the outgoing event will order the execution of a given action. However, other combinations are possible as well, for instance, to be *unobtrusive* (as defined by [11]), the incoming event could be replicated in the output.

Some actions can be conditioned on the occurrence of more than one event. This is the case of *lazy switching handover method*, in which after initiating the handover (receiving the *NetworkSelected(nic)* event) we need to delay the action – or wait for the *TimerOver(delay)* event. To express an action as a consequence of a set of events (e.g., Rule 2) a transducer such as the one in Figure 4 is built.

```
Rule 2:
inst oblig /ProtonPolicies/Obligs/LazyHandover {
  on NetworkSelected(nic) → TimerOver(delay);
  subject /ProtonPMAs/HandoverPMA;
  target t = /ProtonTargets/HandoverExecutor;
  do t.handoff(nic);
  when t.isRAreceived(nic);
}
```

For the sake of simplicity, we disregard the fact that events can arrive without order, and we do not include other possible events before and after the sequence of interest in the model. The symbol "?" represents the TF associated to the *all-events* condition.

Constraints – Constraints are expressed using the *composition* TFFST operation seen in [4], an analogue operation to composition between functions. After all the obligations are represented in a single transducer, the transducer representing constraints should be subsequently composed.

To see how constraints work, let us assume the *InsertHysteresis* example of Rule 3. If we rely only on the plain policy, if a *FadingSignal(nic)* event occurs, the host can fall into the ping-pong effect. One possibility for avoiding this situation is to create the following constraint:

```
Rule 3:
inst oblig /ProtonPolicies/Obligs/InsertHysteresis {
  on FadingSignal(nic);
  subject /ProtonPMAs/HandoverPMA;
  target t = /ProtonTargets/HandoverExecutor;
  do t.ignoreFadingEvent(nic);
  when t.hysteresisPeriod(time);
}
```

The transducer shown in Figure 5 represents this constraint. Computing the *composition* of both transducers produces the solution shown in Figure 6, in which all possible system responses are analysed a priori in the network side.

V. Processes

This section describes the processes related to the policy model. Several tasks have to be performed to generate, deploy, and evaluate the TFFST corresponding to a policy set (see Figure 7). These tasks are: policy translation, conflict resolution, model deployment, context gathering, policy evaluation and tautness function computation.

A. Policy Translation

Policy translation from high-level languages into internal policy evaluation models can be a complex task that needs to be kept simple and ad-hoc in our system. As mentioned above, we must translate high-level policies built with the Ponder policy specification language into the internal policy evaluation model comprised of TFFSTs. The translation process follows the principles presented in Subsection IV-D.

A clear view of the links between objects generated by Ponder tools and the TFFST structure is shown in Figure 8. Ponder distribution [5] was modified to handle the new TFFST structures and support the translation process.

The main challenge of the deployment was the implementation of the TFs associated with the policy conditions. Considering the fact that the PROTON policy model is based on the tools provided by Ponder, the correct approach was to keep its object-oriented approach using target and subject methods to compute TFs.

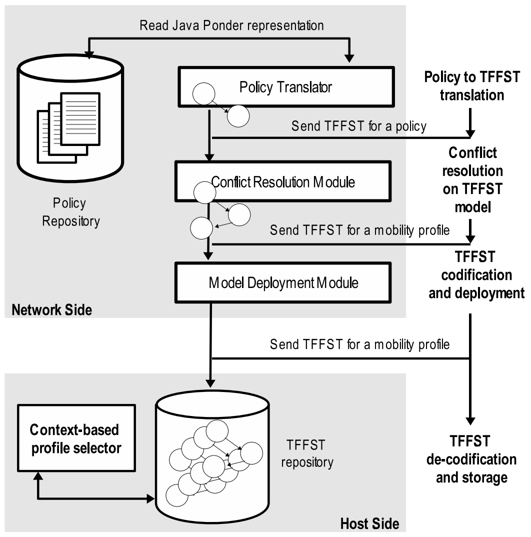

Read Java Ponder representation
Policy Translator
Policy to TFFST translation
Send TFFST for a policy
Conflict resolution on TFFST model
Conflict Resolution Module
Policy Repository
Send TFFST for a mobility profile
TFFST codification and deployment
Model Deployment Module
Network Side
Send TFFST for a mobility profile
Context-based profile selector
TFFST repository
TFFST de-codification and storage
Host Side

Fig. 7. Model Deployment process

Thus, when target or subject methods are called to check a *when* clause, a corresponding method is called at the same time to assign a TF value instead of the boolean value that Ponder assigns to the condition. This method should be developed explicitly, enabling the design of different TF computations depending on the specific parameter (for conditions represented by logic combinations of simple conditions, the TF algebra remains valid).

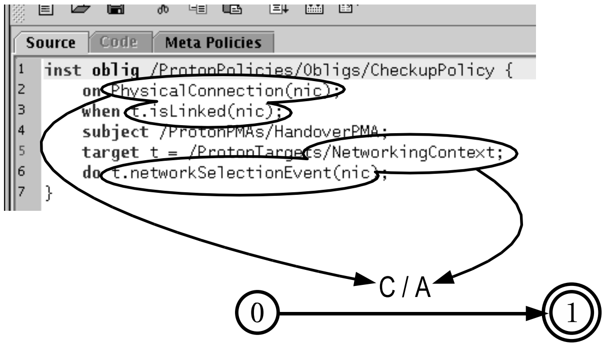
B. Conflict Resolution

An advantage of using transducers to model policies is the rich set of operations available. We can join, intersect, complement, compose and determinise transducers under certain conditions. To build a TFFST that models a set of positive obligation policies, we must build one for each policy and join them using the *union* operation. However, the union of TFFSTs maintains ambiguities and contradictions. Therefore, *determinisation* and *composition* operations must be performed to eliminate these problems.

Determinisation transforms a TFFST into its deterministic and unambiguous version, in fact it also eliminates *static conflicts* between policies.

A TFFST M is *deterministic* if M has a single starting state, if there are no states $p, q \in Q$ such that $(p, \epsilon, x, q, i) \in \Pi$, and if for every state p and event e there is at most one transition (p, τ_d, x, q, i) such that $\tau_d(e)$ is positive.

If a TFFST is deterministic then the process of computing the output actions for a given stream of events ω, can be implemented efficiently. This process is linear in ω, and independent of the size of the TFFST. The determinisation algorithm has two main stages:
Eliminating apparent local conflicts. Local ambiguity may not be such if by analysing the whole transducer, we realise that only one path is possible until the final state. This is the case of the ambiguity shown in *state 0* (see Figure 9(a)). Therefore, outputs are delayed as much as possible.



```
inst oblig /ProtonPolicies/Obligs/CheckupPolicy {
    on PhysicalConnection(nic);
    when t.isLinked(nic);
    subject /ProtonPMAs/HandoverPMA;
    target t = /ProtonTargets/NetworkingContext;
    do t.networkSelectionEvent(nic);
}
```

Fig. 8. Translation Process

Resolving static conflicts. If it is not possible to delay the decision more, the second stage begins. A transition is created for each possible combination between potentially conflictive conditions applying the following criteria: although an event satisfies two conditions, one of these conditions fits more *tautly* than the other. The idea of tautness is represented by the Tautness Functions defined in Subsection IV-C, which can be used to compare orthogonal conditions.

In the output part of the transition, actions and events are arranged following the order given by operators on the input. These operators are in fact part of the output. Later in the process these operators will be eliminated by the *composition* of transducers to apply the given constraints in the system. Figure 9(b) shows the transducer after determinisation.

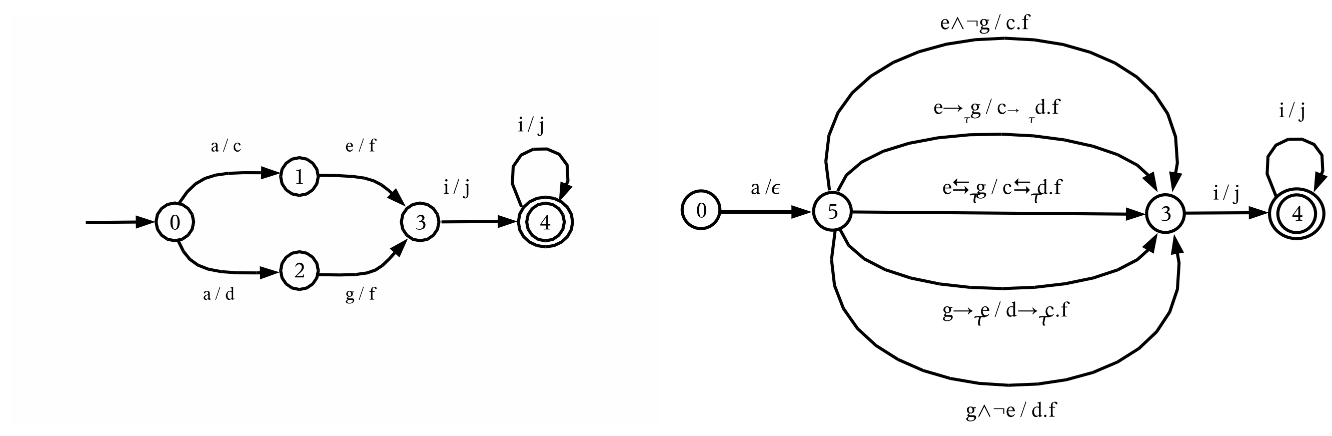
Composition eliminates semantic contradictions (i.e. dynamic conflicts) between the actions. This operation between transducers is equivalent to the composition of any other binary relation: $R_1 \circ R_2 = \{(x, z) \mid (x, y) \in R_1, (y, z) \in R_2\}$.

Thus, the process can be understood as a chain of events where the events and actions in the output of the first transducer are considered to be the input of the second one. The advantage is that the chain process is performed analytically in the network and not on the mobile device.

Thus, if we create a TFFST that replicates all input actions on the output except for those patterns of actions not allowed on the system, and then we compute the *composition* of that transducer with the TFFST policy model, we obtain a transducer that enforces actions without dynamic conflicts. This means actions that must not be performed at the same time, for example two handovers, each one to a different network.

Consequently, conflict resolution is intrinsic to the model. This process not only builds the transducer that models the policies, but also eliminates ambiguities and contradictions between those rules. The main steps are:

1) Compute the union of all transducers representing *rights* and *obligations*.
2) Substract the transducers representing *prohibitions* and *dispensations*.
3) Compose the resulting transducer for each constraint transducer.
4) Determinise the resulting transducer to solve conflicts.


a / c
e / f
i / j
i / j
a / d
g / f
0
1
2
3
4
$e\wedge\neg g$ / c.f
$e \rightarrow_\tau g$ / $c \rightarrow_\tau d$.f
$e \leftrightarrows_\tau g$ / $c \leftrightarrows_\tau d$.f
$g \rightarrow_\tau e$ / $d \rightarrow_\tau c$.f
$g\wedge\neg e$ / d.f
a / ϵ
i / j
i / j
0
5
3
4

(a) Before determinisation (b) After determinisation

Fig. 9. Determinisation process.

Determinisation and composition operations are extensions to the algorithms developed for Predicate-augmented Finite State Transducers [10]. Baliosian et al. presents a detailed explanation of these extensions in [4].

C. Model Deployment

After policy translation, conflict resolution and TFFSTs composition, the final set of TFFSTs for every mobility profile is built (everything so far happens on the network side). Thereafter, the TFFSTs need to be sent to the repository in the mobile host. The Model Deployment Module together with the Policy Master, are responsible for the installation process.

The TFFSTs are kept in a repository, and loaded jointly with the mobility profile according to the reception of a macro-event, e.g., LowAutomobileVelocity(), see Figure 7.

D. Context Gathering

Table III shows the relation between context fragments and the correspondent CML component responsible for polling or retrieving the data. The process of gathering the Networking Context has three steps. The first task is done by Sentinels and Retrievers in the CML, and consists of polling context data (i.e. Collected dataset). Then, the resulting information is filtered according to local rules – the Aggregated dataset is the result of this step. Finally, the CML components maintain a snapshot (i.e. Networking Context) of the context fragments to evaluate policies and generated events (see Table III) when a relevant change in this information occurs.

E. Policy Evaluation

Policy evaluation occurs in the TFFST model. As we mentioned earlier, the computational load of deterministic transducers does not depend on the size of the transducer but rather only on the length of the input. This is possible because the computation consists of following the only possible path corresponding to the input represented by an *epoch* or window of events, which are considered simultaneous for the purpose of detecting dynamic conflicts.

When evaluating the epoch, the transducer performs two tasks: it checks the current epoch and decides if it contains a relevant event pattern in order to decide whether or not to accept it; then it produces a sequence of actions for every accepted epoch, which is sent to the Policy Enforcement component.

F. Tautness Function Computation

A fundamental process in the deployment of TFFST models is the appropriate computation of tautness functions. Our prototype handles each parameter individually with the common idea of expressing the probability of a condition.

For example when computing a condition (related to bandwidth) such as the one below:

```
...
when t.effectiveBW([nic A]) <
     t.effectiveBW([nic B]);
...
```

If *nic A* is connected to a hotspot and *nic B* uses Vodafone's GSM/GPRS network, considering the maximum data rates presented in Table I, and assuming their values have uniform distributions, when we evaluate the condition to *true*, its tautness function is:

$$\frac{144kb/s}{11Mb/s \times 1000} \times 0.5 = 0.00654$$

A value as close to zero as this one means a *very strong condition*. Hence, it is *very unlikely* that this situation will occur and the manager must have had a very good reason to specify a policy with this condition. Therefore, during the determinisation process this condition will have a high priority. Nevertheless, at runtime each TF value will be pondered according to the user profile.

TABLE III
CML COMPONENTS, CONTEXT FRAGMENTS, AND GENERATED EVENTS.

Context fragment	Component	Event

Layer 1 connectivity	AttachedSentinel	PhysicalConnection(nic)
		PhysicalDisconnection(nic)

Signal strength	SignalSentinel	FadingSignal(nic)

Layer 2 connectivity	LinkedSentinel	LinkConnection(nic)
		LinkDisconnection(nic)

Layer 3 connectivity	RouterAdsSentinel	NetworkConnection(nic)
		NetworkDisconnection(nic)

Handover latency	HandoverRetriever	no generated events
Logical position	LogicPositionSentinel	ChangeLogicPosition(address)

Physical position	PositionSentinel	ChangePosition(position)
		ContextChangeTransition()

Velocity	VelocitySentinel	PedestrianVelocity()
		LowAutomobileVelocity()
		HighAutomobileVelocity()
		HighSpeedVelocity()

Direction	DirectionSentinel	ChangeDirection(direction)
Network traffic	TrafficSentinel	ChangeTraffic(nic)
User profile	UserRetriever	ChangePreference(preference)
Ongoing applications	FlowsSentinel	NewDataFlow(trafficType)
Network charac.	NetworkRetriever	no generated events
App. characteristics	ApplicationRetriever	no generated events
Network structure	InfrastructureSentinel	NearbyAccess(positionArray)

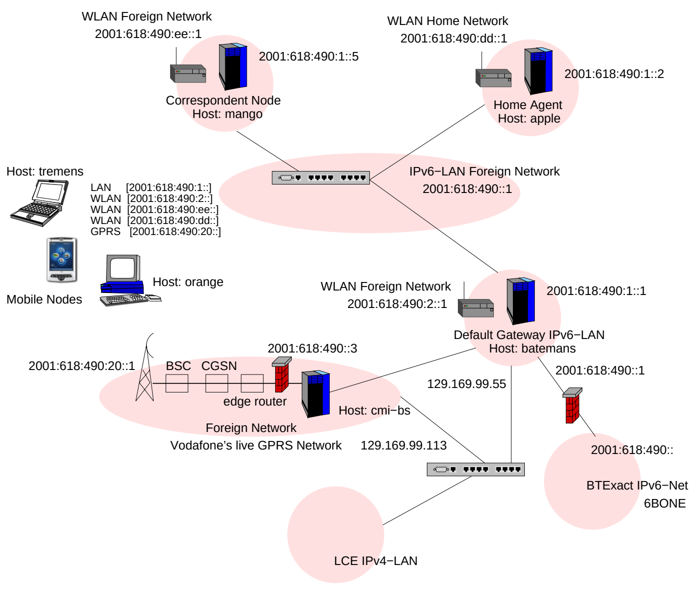


Fig. 10. LCE-CL setup enables seamless inter-network roaming.

VI. Policy Enforcement

As the Policy Master moves through the selected path in the TFFST, it evaluates conditions and generates actions to be enforced by the *executors*. The executors can play the role of *subject* or *target* in the policy [6]. For example, the executor *HandoverExecutor* plays the role of target, and is responsible for executing methods to evaluate conditions and perform actions.

There are two type of actions: internal and external. The former (e.g. *networkSelectionEvent*) are performed within PROTON. The latter, for example *executeUpwardHandover*, occurs between PROTON and the mobile host (see Figure 11), and these are executed by the Control Interface that lies between the network layer and the mobility management sub-layer (i.e. MIPv6 module). The interface controls the incoming router advertisements from different access networks and it executes the corresponding actions (received from the Handover Executor, according to the Networking Context and the TFFST).

Actions are associated with the different stages of the handover process. The Control Interface runs scripts based on IPv6tables (see example below), which build appropriate rules to inhibit automatic handovers (by filtering router advertisements) and enable handovers according to the network selection process. These scripts also set timers considering context (e.g., mobile host velocity) and execute the most convenient handover mechanism.

```
wlan_gprs)
  echo Setting MIPL preference to handoff to GPRS [sit1]
  mipdiag -i sit1 -P 3
  mipdiag -i eth1 -P 2
  mipdiag -i eth0 -P 1
  echo PROTON: ACCEPT RAs from GPRS [sit1]...
  ip6tables -D INPUT -i sit1 -j DROP
  echo Waiting for 5 seconds [soft handover]
  sleep 5
  echo PROTON: DROP RAs from WLAN [eth1]...
  ip6tables -A INPUT -i eth1 -j DROP
  echo PROTON: done...
;;
```

This example enforces a set of policies that suggest the execution of a soft handover from a hotspot to the cellular system with a waiting time of 5s (this period is based on context), during which the mobile host is listening to both interfaces, executing a sort of method equivalent to lazy cell switching in horizontal scenarios.

VII. Experimental Results

A. Testbed

To closely emulate the next generation (4G) integrated networking environment, our experimental testbed setup consists of a tightly-integrated, Mobile IPv6-based GPRS-WLAN-LAN testbed as shown in Figure 10. The cellular GPRS network infrastructure currently in use is Vodafone UK's production GPRS network. The WLAN access points (APs) are IEEE 802.11b APs. Our testbed has been operational since March 2003, and results showing how we optimise vertical handovers are detailed in [12].

For access to the 4G integrated network, mobile hosts (e.g. laptops) connect to the local WLAN network and also simultaneously to GPRS via a Phone/PCCard modem. The mobile host's MIPv6 implementation is based on that developed by the MediaPoli project [13], chosen for its completeness and open source nature.

A router in the lab acts as an IPv6/IPv4 tunnel end-point to the BTExact's IPv6 network. There is an IPv6 access router (Home Agent) for the lab's fixed-internal IPv6-enabled network and also for internal WLANs (shown in Figure 10).

We used the testbed to evaluate PROTON in the most common 4G scenarios. For example, assistance to mobile users while offering seamless service continuity between the different access technologies by minimising the impact of vertical handovers. The aim is to observe how the policy model

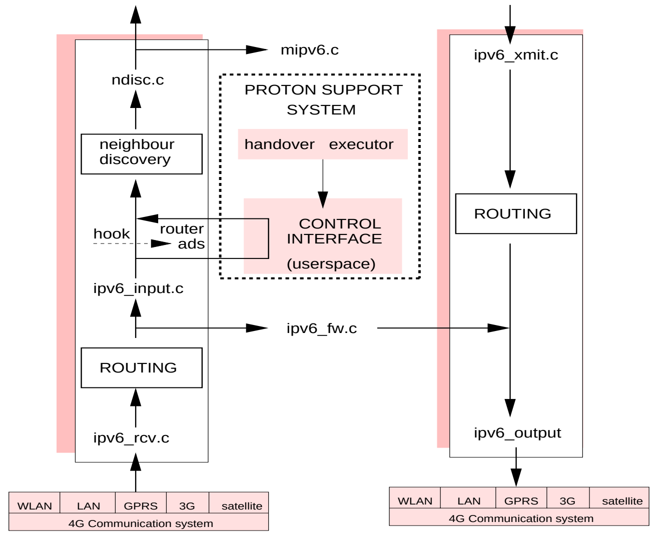


Fig. 11. Handover Executor Implementation.

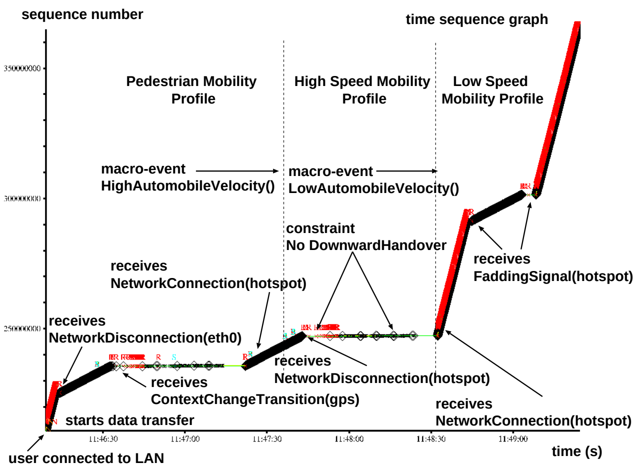


Fig. 12. Testing PROTON in a case scenario.

and the system itself respond to the determined conditions and execute handover-related decisions.

B. Evaluation Examples

We evaluate PROTON simulating a real 4G situation using the LCE-CL testbed (see Figure 12). To do this, we installed PROTON in a multimode device (a Toshiba Satellite laptop) that can access multiple wireless technologies (e.g., IEEE 802.11b, IEEE 802.11a, IEEE 802.11g, GSM/GPRS, and Ethernet) and forced a sequence of events that triggers the evaluation of certain policies, and the execution of the corresponding actions.

Imagine Nancy in her office using her PROTON-enabled laptop, she starts downloading a huge amount of data that she needs for an important business lunch in London's city centre. She decides to leave her office immediately and continue downloading the data on-the-move. When Nancy disconnects her laptop from the local network, the first PROTON-event is generated: *NetworkDisconnection(eth0)*. This triggers policies and actions, the laptop connects to the WiFi available in the building and continues downloading the data without any disruptions.

When she leaves the building, the *PositionSentinel* cannot read the data from the indoors location system (e.g. bat system [14]), and the second event is generated: *ContextChangeTransition(gps)*. The mobile device starts reading its location using the GPS receiver, and it connects to the available cellular system (e.g. Vodafone's GSM/GPRS network). As Nancy approaches her car, PROTON detects a nearby hotspot – *NetworkConnection(hotspot)*. It seamlessly evaluates the appropriate set of policies and decides to use this broadband network.

She starts driving on the highway toward the city centre, and as the car accelerates PROTON uses the GPS receiver to monitor the velocity and generates a macro-event: *HighAutomobileVelocity()*. The process in Figure 13 occurs and the corresponding TFFST is loaded. PROTON connects to the GPRS network when the hotspot connectivity is lost, and while Nancy is driving on the highway no downward handovers are allowed – because of the constraint *No DownwardHandover* specified in the *HighAutomobileVelocity* mobility profile and built in the corresponding TFFST. Thus, she stays connected to the GPRS system.

She reduces the speed as she reaches the traffic areas in the city centre. This situation is detected by the *VelocitySentinel* and the macro-event *LowAutomobileVelocity()* is sent. Another mobility profile is loaded and the *NetworkConnection(hotspot)* event received. Autonomously, PROTON evaluates the TFFST and decides to continue with the data transfer using the available hotspot. A few minutes later, the signal from the current access point starts fading and the event *FadingSignal(hotspot)* is generated. PROTON changes its attachment point without disruptions; it uses the most appropriate execution method, initiation time, and adapts itself to the new QoS conditions exploiting its policy model and Networking Context dataset. She arrives to her final destination, PROTON connects to the restaurant's hotspot to download the last few bits of data, and Nancy starts her meeting.

The described scenario was simulated using the LCE-CL testbed and the expected results observed. User experience improves because they can continue their tasks on-the-move. Furthermore, system performance increases using this ubiquitous access network. Seamless roaming between heterogeneous networks was enabled using the policy model, and the resulting overhead was acceptable for this type of environment.

C. Scalability Issues

A possible disadvantage of TFFSTs is the high order of their algorithms and the size of the final transducer. In praxis, considering the heuristics in PROTON strategies, we can control the internal TFFST model and keep its size within acceptable limits.

As described in Section III, not every context fragment matters in every situation. In PROTON, important fragments are selected according to the *mobility profiles*. Hence, a

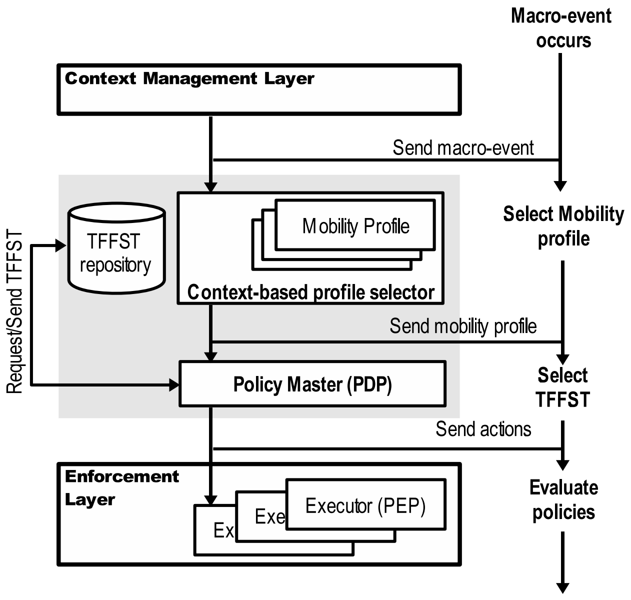


Fig. 13. TFFST Selection on Macro-events

WLAN ⇒ GPRS	Mean	Std. Dev.	Min.	Max.
Detection time (t_d)	808	320	200	1148
Configuration time (t_c)	1	0	1	1
Registration time (t_r)	2997	416	2339	3649
Total handover latency (t_h)	3806	327	3323	4438

GPRS ⇒ WLAN	Mean	Std. Dev.	Min.	Max.
Detection time (t_d)	2241	968	739	3803
Configuration time (t_c)	1	0	0	1
Registration time (t_r)	4654	1698	2585	7639
Total handover latency (t_h)	6897	1178	5322	8833

LAN ⇒ GPRS	Mean	Std. Dev.	Min.	Max.
Detection time (t_d)	1168	460	347	2070
Configuration time (t_c)	1	0	1	1
Registration time (t_r)	3307	585	2299	4759
Total handover latency (t_h)	4476	520	2806	5107

GPRS ⇒ LAN	Mean	Std. Dev.	Min.	Max.
Detection time (t_d)	2058	1030	1	3257
Configuration time (t_c)	1	0	1	1
Registration time (t_r)	4466	1449	2357	7183
Total handover latency (t_h)	6525	1229	4011	8197

Fig. 14. Latency partition for vertical handovers during a TCP transfer.

transducer is built for each profile, reducing the maximum size of each TFFST and avoiding processing overhead and minimising storage space in the mobile device.

Figure 13 shows the TFFST selection process at the host side. It can be seen as a hot-reprogramming of the device to optimise its behaviour considering each scenario.

The current version implements four different mobility profiles according to the context fragment *Velocity* that produces the following macro-events: PedestrianVelocity, LowAutomobileVelocity, HighAutomobileVelocity, and HighSpeedVelocity. Every time that one of these macro-events is generated, the corresponding mobility profile is loaded.

Experiments in different scenarios showed that the number of transitions for each mobility profile's TFFST was dependent of the quantity of relevant context fragments, possible events, and applied constraints. These variations in numbers respond to the following facts:

- *At lower speeds more context fragments can be considered to take decisions, increasing the number of transitions.*
- *At higher speeds more constraints can be applied to the policy model, reducing the number of transitions.*
- *At higher velocities, fewer events are relevant for making decisions, decreasing the amount of transitions.*

From experiments, we observed that the number of transitions for pedestrian speed is much higher (around 9000 transitions) compared to the rest of the mobility profiles: for automobile velocity the number is around 2000 and approximately 100 for high speed profiles. Nevertheless, these numbers do not affect the evaluation process (host side) and it only increases computational cost in the network.

We also need to consider the memory space needed to store TFFSTs in the mobile device. In PROTON, transitions are represented by simple objects of a size equal to 333 bytes. A TFFST can be seen as a vector of transitions that in the worst scenario (pedestrian profile) will only require 332KB of memory space. Therefore, storage of TFFSTs does not represent a scalability constraint.

TABLE IV

RUN-TIME PERFORMANCE FOR POLICY EVALUATION.

Mobility profile	Out-degree	Evaluation time (ms)
Pedestrian	3165	396
Automobile	316	108
High Automobile	39	24
High Speed	39	24

D. Run-time Performance

For the run-time performance, the significant times are those on the host side. There are three main stages to consider: *context gathering*, *policies evaluation* and *policy enforcement*.

Context gathering – Updating the Networking Context dataset implies polling files, the operating system, and peripherals connected to the host. The total time required to update all the components is between 200ms and 300ms. How often the CML updates the Networking Context dataset well depends on the mobile host's velocity. We propose the following values: for pedestrian speed every 4s to 8s (this means every 5m to 10m). At higher velocities the dataset is updated every second.

Policy evaluation – The evaluation time is linear to the input size, and it does not depend on the transducer's size. However, the final evaluation time does depend on the maximum number of outgoing transitions belonging to an state in the transducer (transducer's out-degree). In each stage of the deterministic transducer evaluation, the valid transition must be selected among all the transitions associated to the state. Thus, the maximum evaluation time for a set of events is: $E \times Max_i\,(T_i)$, where E is the amount of events in the input and T_i the number of outgoing transitions for the state i.

Table IV shows the evaluation time for an input of two events using each of the mobility profiles. We also show the out-degree of the mobility-profile's TFFST.

Policy enforcement – All the actions must occur during the handover latency. Tables in Figure 14 show the latencies for the most common inter-networks handovers, measured using the LCE-CL testbed. Within this period (i.e. t_h) the Enforcement Layer needs to execute an average of five actions associated to the different stages of the handover process.

VIII. Related Work

The IP technology growth explosion –mainly due to the popularity of Internet services– brings to the fore *network convergence* as an immediate challenge. Thus, considering an IP core network as the next generation architecture, Mobile IP [15] represents a *de facto* solution for mobility management in this environment. However, handover-related decisions such as networks detection, network selection, execution methods, and handover initiation are outside the scope of the current specification. Mobile IPv6 only deals with networking issues to enable mobility in networked environments.

Thus, a number of strategies to perform effective handovers in heterogeneous systems have been explored since 1996, when the concept of *Overlay Networks* in wireless environments first appeared in the Daedalus project [16]. As part of this work, a pioneer policy-based solution for mobility support was proposed by Wang et al. [17]. This solution supported network selection and handover execution processes; however, its policy model was mainly focused on the network selection using cost functions to ponder input data such as cost, bandwidth, and charge model. The authors mentioned that offering full assistance will result in an excessive increase in complexity; for this reason, we argue that it would be better to avoid the use of cost functions, due to their computing constraints and lack of flexibility.

This achievement was followed by other policy-based approaches to tackle different handover-related problems such as data-flow based selection of the most appropriate access technology [18]. Although many solutions have been proposed to solve inter-system handover challenges, it is only lately that complete mobility support solutions have been envisaged.

Because of the popularity of IP-based services, the completeness of the solution ponders its compatibility with this protocol. In addition, a complete solution should be pro-active and reckon context in the decisions while offering full support. Finally, a more appropriate solution should be feasible to deploy, this last characteristic is closely related to the entities involved in the deployment, which can either be the mobile host, the network, or both. Following this criteria, we compare PROTON to some closely related approaches. Table V lists the most relevant mobility management solutions in recent years.

Recently, Vardalachos et al. [19] initiated the development of a network-assisted policy management system for hybrid networks, not specially focused on 4G systems at the beginning. However, this work continued as part of the IST project CONTEXT where Murray et al. [20] described a context-aware system to control handover initiation in next generation networks. The main difference with these proposals is that they are network-based and they affect the network infrastructure. Furthermore, as they require network data, they are not as dynamic as the mobile-based approach – in which decisions are made just considering immediate context.

An extension of this work was published by Yang et al. [21], adding scalability problems to the system by introducing mobile agents to enable service delivery between the network and the clients.

In [22] a more alike solution is presented, Fikouras et al. describe POLIMAND, a policy-based MIP handover decision method. The policy model in POLIMAND considers only link layer data (basically signal strength), which prevents the solution from offering full support. It does not assist users during network selection or adaptation processes, mainly because of the lack of inputs from other layers or even physical context.

Other proposals such as Murray et al., [23] Makela et al., [24], and Chan et al., [25] explore the use of other decision methods. We believe that a policy-based approach is sufficient to handle complexities in 4G systems. Consequently, other schemes such as fuzzy logic and neural networks are far too complex and they add undesired overhead to the decision process.

Furthermore, most situations in the handover process can be modelled using a linear system that receives precise inputs – in this scenario the use of fuzzy logic becomes excessive. Finally, dynamics in 4G environments demands agile and appropriate decisions, and not necessarily the best one. Thus, complex decision models are not always the best approach to enable mobility support in 4G networks.

Our solution uses a policy-based decision schema following the IETF PCIM specification [7]. The policy evaluation model builds on the concept of *Finite State Transducers* (FSTs), and it is intended to provide both a fast evaluation model and effective conflict resolution algorithms. We deploy extensions to algorithms developed for natural language processing [10]. However, these methods [4] were adapted to mimic strategies that emerged from previous research on static conflicts [26] and dynamic conflicts [11]. Additionally, a new metric called Tautness Function is used to abstract technology-dependant conditions and context variables [4].

The most common method to resolve conflicts is to explicitly assign priorities to policies and decide on the one with higher value. A more complex method is the *goal-oriented strategy*, which consists of assigning priorities to every possible system state and moving on to the state with higher priority [27].

The proposed conflict resolution module prioritises conditions automatically. This strategy resolves policy conflicts in a simple manner and it is powerful enough to solve most of the situations without human intervention.

Dunlop et al. [28] use *conflicts databases*. Their work is related to ours in the sense that both solutions consider every possible conflict beforehand. They only detect conflicts while we perform detection and resolution. Furthermore, conflicts databases can prove to be unsuitable for our purposes whereas Finite State Machines represent a light-weight solution that is

TABLE V
PROTON SOLUTION STANDS OUT FROM PREVIOUS APPROACHES BECAUSE (1) IT PROVIDES FULL SUPPORT TO 4G MOBILE USERS, (2) IT IS CONTEXT-AWARE, AND (3) CLIENT-BASED.

Authors	Scheme	IP-based	Context	Decision	Initiation	Selection	Execution	Adaptation
H. Wang et al. (1999)	Policy based	YES	NO	terminal	NO	YES	YES	NO
J. Makela et al. (2000)	Neural networks	NO	NO	terminal	YES	NO	NO	NO
P. Chan et al. (2001)	Fuzzy logic	NO	NO	terminal	YES	YES	NO	NO
K. Jean et al. (2003)	Policy based	NO	YES	network	YES	NO	NO	NO
N. Vardalachos et al. (2003)	Policy based	NO	NO	network	YES	NO	NO	NO
K. Yang et al. (2003)	Policy based	NO	YES	network	YES	NO	NO	NO
N. Fikouras et al. (2003)	Policy based	YES	NO	terminal	YES	NO	YES	NO
K. Murray et al.	Policy based	NO	NO	terminal	NO	YES	NO	NO
K. Murray et al.	Fuzzy logic	NO	NO	terminal	YES	NO	NO	NO
P. Vidales et al. (2004)	Policy based	YES	YES	terminal	YES	YES	YES	YES

more feasible to deploy in mobile devices.

In sum, PROTON differs from previous schemes in the following concepts:

- PROTON is designed considering high-dynamic and complexity in 4G environments. It is a mobile-based solution, however, most of the computational load is kept on the network.
- PROTON is a a context-aware system that considers not only network conditions but also other context fragments (e.g., physical environment and user preferences), which are equally important to properly solve handover-related situations.
- PROTON offers complete mobility support, this is a key advantage in 4G mobile systems. Decisions before, during, and after handover execution will improve mobile users' experience.
- PROTON is entirely mobile-based; however, network knowledge can be transfered to the wireless device through the Model Deployment process.
- PROTON attempts to implement an autonomic solution for 4G systems.

IX. Further Research

An interesting aspect of PROTON is the concept of deploying TFFSTs considering other aspects such as operator's business model, strategies, or even mobile device characteristics, and not only mobility aspects as evaluated in this paper. The behaviour of the mobile device is driven by the TFFST evaluation, thus by implementing different automata we can explore more complex system responses.

Before deploying PROTON, adjustments must be made to the prototype. The accuracy of the policies requires further evaluation, and based on the knowledge obtained we can adjust the policy rules. Although we showed that PROTON is capable of offering full support to 4G mobile users, better results can be obtained by making the proper modifications.

The PROTON prototype is an early implementation and there are many performance issues that need to be solved. For example, the communication protocol used to install TFFSTs in the mobile device needs to be improved. The internal representation of TFFSTs can be smaller and faster evaluation algorithms can be deployed.

The TF computation strategy needs further tuning and testing. Non-linear translations of conditions' parameters will probably turn out to be more accurate and meaningful than the linear expressions used in the current version of the system. Also, ponderers based on user preferences need to be adjusted to work better in combination with TFs.

PROTON supports the aggregation of new sets of policies to assist users in other tasks. For example, we have considered the implementation of a policy set for security in such ubiquitous environments [29]. Policies for data adaptation [30] are essential to achieve seamless roaming; this is an interesting research topic that needs further work.

We planned to use PROTON to feed registered applications (i.e. consumers) with context information. The development of an API to enable the deployment of novel context-aware 4G services and applications could be a future hot topic.

Finally, the completion of the full automation of the system and interfacing of PROTON and external components such as Ponder and network-side elements needs more work.

X. Conclusions

In this paper, we presented PROTON, a policy-based system to support multimode devices. Motivation behind PROTON stems from the fact that future devices will have multimode capability for connecting to different wireless networks. We demonstrated how PROTON can address several issues of future networking, and how it can cope with complexity and dynamics intrinsic to future environments.

As far as we know, PROTON is the first policy-based system that attempts to offer complete mobility support for 4G mobile users. These heterogeneous environments pose challenges that remain open. Using a policy model based on TFFSTs, PROTON helps users in many decisions while hiding the added complexities.

We have also demonstrated that concepts from autonomic computing can be applied to the design of novel solutions that brings us closer to the answer of open networking challenges such as seamless roaming among heterogeneous technologies.

This project consolidates the idea of building the policy evaluation model on the network, to enable devices with the capability to deal with complexities while keeping a powerful light-weight solution.

The Networking Context dataset presented is rich enough to allow well-informed decisions while roaming. However, the possibility of using a rich set in such a dynamic environment is empowered by the idea of having three levels of information according to the dynamics of data elements.

PROTON's architecture also reflects the concern of dealing with constraint devices, particularly in a constantly-changing environment. This is the main drive behind dividing PROTON into network- and host-side components. Every module that demands intensive computation work or high storage capacity is located in the network.

The mobile device deals, exclusively, with the evaluation of TFFSTs, a task that does not require much processing. Via the application of novel algorithms for the specification and translation of policies, conflict resolution, and TFFST deployment and installation, we have implemented a complete system that supports seamless roaming in upcoming pervasive networking environments, while representing a light-weight solution that is easy to deploy.

We need powerful and more intelligent solutions to support inter-networking in future communication systems. However, mobile devices and wireless environments will always exhibit strong limitations in terms of memory capacity, processing power, and stability. Hence, the prior resolution of conflicts and TFFST deployment is a very appropriate approach to overcome these constraints.

PROTON has demonstrated the potential of merging concepts of autonomic computing with the design and implementation of a policy-based system, together with a novel evaluation model and efficient conflict-resolution algorithms. The result: a solution that offers full mobility support, hides complexities, enables smart decision-making while roaming, and deals with the intrinsic constraints of 4G environments.

XI. Acknowledgements

The authors would like to thank Calicrates Policroniades and Carlos J. Bernardos for their helpful comments. Pablo Vidales has a scholarship from the Mexican Government through the National Council of Science and Technology (CONACyT) and he is partially funded by the Laboratory for Communication Engineering.

References

[1] R. H. Katz, "Adaptation and mobility in wireless information systems," *IEEE Personal Communications*, vol. 1, pp. 6–17, 1994. [Online]. Available: citeseer.nj.nec.com/katz95adaptation.html

[2] IBM Research Headquarters (manifesto), "Autonomic Computing: IBM's Perspective on the State of Information Technology," October 2001. [Online]. Available: http://www.research.ibm.com/autonomic/overview/elements.html

[3] P. Vidales, R. Chackravorty, and C. Policroniades, "PROTON: A Policy-based Solution for Future 4G devices," in *Proceedings of The Fifth IEEE International Workshop on Policies for Distributed Systems (POLICY 2004)*, June 2004.

[4] J. Baliosian and J. Serrat, "Finite state transducers for policy evaluation and conflict resolution," in *Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)*, June 2004.

[5] "Policy Research Group. `http://www-dse.doc.ic.ac.uk/Research/policies/ponder.shtml`."

[6] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The Ponder Policy Specification Language," in *Proceedings of the Second IEEE International Workshop on Policies for Distributed Systems (POLICY 2001)*, January 2001, pp. 18–39.

[7] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, "Policy Core Information Model, Internet RFC rfc3060.txt," February 2001. [Online]. Available: `http://www.ietf.org/rfc/rfc3060.txt`

[8] G. Fitzpatrick, S. Kaplan, T. Mansfield, D. Arnold, and B. Segal, "Supporting Public Availability and Accessibility with Elvin: Experiences and Reflections," in *Computer Supported Collaborative Work: the Journal of Collaborative Computing*, October 2000, pp. 15–51.

[9] E. Roche and Y. Schabes, "Finite-state language processing," MIT Press, Cambridge, Massachusetts., Tech. Rep., 1997.

[10] G. van Noord and D. Gerdemann, "Finite state transducers with predicates and identities," *Grammars*, vol. 4, no. 3, pp. 263–286, December 2001.

[11] J. Chomicki, J. Lobo, and S. Naqvi, "Conflict resolution using logic programming," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 1, pp. 245–250, Jan/Feb 2003.

[12] R. Chackravorty, P. Vidales, I. Pratt, and J. Crowcroft, "On TCP Performance during Vertical Handovers: Experiences from GPRS-WLAN Integration," in *Proceedings of The Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, March 2004.

[13] "Mobile IP for Linux (MIPL) Implementation by HUT Telecommunications and Multimedia Lab, `http://www.mipl.mediapoli.com`."

[14] "AT&T laboratories cambridge. `http://www.uk.research.att.com/bat`."

[15] D. B. Johnson, C. E. Perkins, and J. Arkko, "Mobility Support in IPv6 (RFC 3775)," June 2004. [Online]. Available: `http://www.ieft.org`

[16] "Daedalus wireless research group. `http://daedalu.cs.berkeley.edu/`."

[17] H. J. Wang, "Policy-enabled handoffs across heterogeneous wireless networks, Tech. Rep. CSD-98-1027, 23, 1998.

[18] K. Lai, M. Roussopoulos, D. Tang, X. Zhao, and M. Baker, "Experiences with a mobile testbed," in *Proceedings of The Second International Conference on Worldwide Computing and its Applications (WWCA'98)*, March 1998. [Online]. Available: citeseer.nj.nec.com/lai98experiences.html

[19] N. Vardalachos, J. Rubio, A. Galis, and J. Serrat, "A Policy Management System for Hybrid Networks," in *Proceedings of The London Communications Symposium*, 2002.

[20] K. Jean, K. Yang, and A. Galis, "A policy based context-aware service for next generation networks, IEE the Eighth London Communication Symposium," October 2003.

[21] K. Yang, A. Galis, and C. Todd, "Policy-driven mobile agents for context-aware service in next generation networks," in *Proceedings of IFIP Fifth International Conference on Mobile Agents for Telecommunications (MATA 2003)*, October 2003.

[22] S. Aust, N. A. Fikouras, , D. Protel, C. Gorg, and C. Pampu, "Policy Based Mobile IP Handoff Decision (POLIMAND), Internet Draft, draft-iponair-dna-polimand-00.txt, Work in Progress," October 2003. [Online]. Available: `http://www.ietf.org/internet-drafts/draft-iponair-dna-polimand-00.txt`

[23] K. Murray, R. Mathur, and D. Pesch, "Intelligent access and mobility management in heterogeneous wireless networks using policy," in *Proceedings of the First ACM International Workshop on Information and Communication technologies*, 2003, pp. 181–186.

[24] J. Makela, "Handoff Decision in Multi-Service Networks," in *Proceedings of the Eleventh IEEE International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC 2000)*, London, UK, September 2000.

[25] P. Chan, "Mobility Management Incorporating Fuzzy Logic to Heterogeneous IP Environment," IEEE Communications Magazine, 2001.

[26] E. Lupu and M. Sloman, "Conflicts in policy-based distributed systems management," *IEEE Transactions on Software Engineering*, vol. 25, no. 6, pp. 852 –869, Nov/Dec 1999.

[27] J. Kephart and W. Walsh, "An artificial intelligence perspective on autonomic computing policies," in *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, 2004, pp. 3–12.

[28] N. Dunlop, J. Indulska, and K. Raymond, "Dynamic conflict detection in policy-based management systems," in *Proceedings of the Enterprise Distributed Object Computing Conference (EDOC'02)*, 2002, pp. 15–26.

[29] B. Dragovic and J. Crowcroft, “Context-Adaptive Information Security for UbiComp Environments,” in *Proceedings of the Second UK-UbiNet Workshop: Security, trust, privacy and theory for ubiquitous computing*, May 2004.

[30] C. Policroniades, R. Chakravorty, and P. Vidales, “A data repository for fine-grained adaptation in heterogeneous environments,” in *Proceedings of the Third ACM international workshop on Data engineering for wireless and mobile access*. ACM Press, 2003, pp. 51–55.