Home Network Management Policies: Putting the User in the Loop

D. Pediaditakis, A. Gopalan
Department of Computing,
Imperial College London,
London, United Kingdom
{d.pediaditakis, a.gopalan}@imperial.ac.uk

N. Dulay, M. Sloman
Department of Computing,
Imperial College London,
London, United Kingdom
{n.dulay, m.sloman}@imperial.ac.uk

T. Lodge
School of Computer Science
University of Nottingham
Nottingham, United Kingdom
txl@cs.nott.ac.uk

***Abstract*—Home networks are becoming increasingly complex but existing management solutions are not simple to use since they are not tailored to the needs of typical home-users. In this paper we present a new approach to home network management that allows users to formulate quite sophisticated "comic-strip" policies using an attractive iPad application. The policies are based on the management wishes of home users elicited in a user study. Comic-strip policies are passed to a Policy engine running on a new Home Network Router designed to facilitate a variety of management tasks. We illustrate our approach via a number end-to-end experiments in an actual home deployment, using our prototype implementation.**

I. Introduction

Home networks have become an important part in the everyday life of millions of people. It is estimated that during the 3rd quarter of 2011 there were about 581 million broadband subscribers worldwide, and this number is constantly growing [1]. Despite this growth, little work has been done for making these networks easy to manage for the home user. Emerging applications, services and technologies continues to make management difficult, particularly the bewildering array of management interfaces and APIs.

Home networks differ from enterprise ones at many levels. Perhaps, most importantly, typical users are not well trained administrators. They do have the required technology but in many cases don't wish or know how to use it. Where they have some fundamental understanding, it is often in their own terms which is a total different viewpoint from traditional networking ([2], [3], [4], [5], [6]). A second area of difference is that home networks are rarely installed wisely, resulting in chaotic deployments ([7]) which exhibit very poor connectivity characteristics [8]. Bandwidth bottlenecks usually reside at the edges of the Internet ([9]), which is also true for home networks ([10], [11], [12]). Therefore, in a multi-user environment of shared resources it is sometimes necessary to regulate their usage when they are scarce.

Recent studies have proposed new ideas for configuring and managing home networks ([13], [14], [15], [16]) while a few have developed prototype implementations ([17], [18], [19]). Nevertheless, most of the aforementioned works fail to provide an intuitive management model for users, nor do they promote the decoupling of configuration semantics from device and network specific details. There is a clear need for new home network management solutions that address such concerns.

To this end, in this paper, we present a novel policy based approach for home network management that has been driven from user-level requirements elicited from a small group of real users (not researchers or students). In our approach users compose "comic strip" policies on an iPad. Policies are typically defined in terms of actual home users and devices that model ownership of devices. Among the policy actions requested by our users are traffic prioritisation, device blocking and multiple methods for notifications. Actions can be one-off or event-triggered. Event-triggers are typically based on network events (visiting a particular website) or bandwidth usage. Time-constrained policies are also popular.

One of the major contributions of the present work is the fact that policies operate at a higher semantic level. They do not deal with the details of the networking infrastructure and at the same time they can reflect the user needs. This makes easier both the design of a management system but also facilitates advanced reasoning about network's condition and usage. In order to support such an intuitive management model we designed a runtime environment which provides the appropriate abstractions for certain aspects of network monitoring and configuration. The details of the runtime have been discussed in our previous works ([20], [21], [22]). This paper builds upon our existing efforts and completes the design of a closed loop management system for home networks.

The rest of this paper is organised as follows. Section II describes the overall architecture of our system, introducing the types of policies identified and supported, and describes in depth the individual components of our design. Our prototype implementation is presented in section III, which is then used to experimentally evaluate our system in a real home network deployment in section IV. Section V compares the work with related work in the area of home network management. Finally, section VI concludes the paper and outlines our future work directions.

II. System Overview

A. Overall Architecture

In this section we present the overall architecture of our framework and explain the functionality of its components,

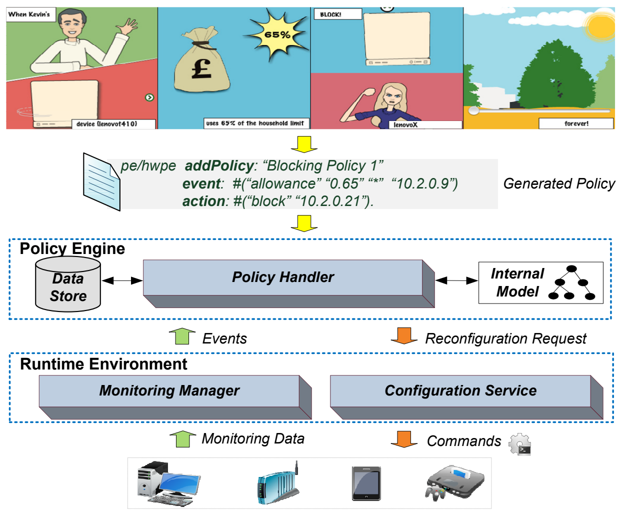

Fig. 1. Overall system's architecture and execution pipeline

illustrated in Fig. 1. Users interact with a "comic strip" iPad application to compose stories that reflect their management requirements. When they submit a new story the application transforms it into a textual representation of the equivalent policy and supplies it to *policy engine* running at the home gateway/router. The engine checks and validates the submitted policy against the running system. If it is not found to contain errors or inconsistencies it gets activated. The specified policy actions can alter the configuration of the network either statically in an one-off manner, or in a dynamic fashion responding to monitoring events. The policy engine maintains the overall state of the network to facilitate validation and to provide feedback to the user.

B. User Interface Design

Users interact with the management system via an iPad which encourages touch-based interaction. iPad apps were looked upon favourably by our group of users for their ease of use and form factor. The policy application that was developed for users prompts them to compose"comic strips" (see Fig. 2) by selecting, combining and configuring pre-defined tiles that correspond to policy elements. Each strip represents a policy. The policies that we support were not based on our ideas for what policies should be provided. Rather they are based on what users asked for. In many cases this was frustrating as it showed a lack of interest from users to many of the policies that we thought they would want (e.g. explicit firewall configuration, port-forwarding, bandwidth allocation etc.). On the other hand, some of the features they wanted required a lot of effort to implement, especially the dynamically applied configuration changes. Currently, most policies take the following form:

- **Tile 1**: when device D_x of user U_x
- **Tile 2**: meets certain usage criteria (event E)
- **Tile 3**: between times t_a, t_b of weekdays W_d
- **Tile 4**: perform Action A on [device D_y of] user U_y

Hosts are mapped 1-1 to users during the device registration phase. Before a device is allowed to connect to the network, the management interface requests a few details from the user (e.g type of device and owner). New users can be registered in a similar way through a separate menu screen. All user to device mappings are static but can be changed at any point.

In Fig. 2 the policy is: "When Kevin's mobile phone accesses Ebay, or Facebook, or YouTube web pages, between 20:00 - 23:29 on any weekday then send Hazel a message on Twitter". This is just one type of policy that users can formulate. Policies can use any of the devices / users / usage events / day-time ranges / actions defined for their home network. Usage events are detected by the underlying monitoring infrastructure (see later) which can be any of the following types: usage of a specific device, access to specific Internet sites (or its negation), and usage of a given percentage of home's total upload and/or download allowance.

The actions which are currently supported fall within three categories: prioritisation, access control and notification (via Email, SMS, Growl, Twitter, Facebook). The first two types of actions may be applied and stay active for a given amount of time (minutes, hours, days) during the date-time ranges that a policy is scheduled to be active. Users can also disable, re-activate, store, remove and maintain sets of "comic strips".

C. Home Network Policies

The "comic strip" stories described in the previous section are visual representations of a rich set of policies which our system supports. They are converted into text form and then forwarded to the policy engine for enforcement. This conversion is based on device ownership information of users that is stored in a relational database running on the home router. Whenever a change is detected in user-device mappings at runtime, the relevant device-level policies are updated, preserving the overall consistency. The syntax of device level policies is:

```
policy:= [Schedule: periodExp]
         [Event:  eventExp, subjectExp]
          Action: actionExp [, subjectExp]
         [Duration: hours(h)  [, mins(m)] ]
periodExp:= (hh:mm, hh:mm, {weekday})
weekday:= {Sun|Mon|Tue|Wed|Thu|Fri|Sat|*}
eventExp:= Visits ([!]{url}) |
          Used |
          Allowance (percentage,(up|down))
subjectExp:= {ipaddr} | any
actionExp:= {Notify (service, "details") |
             Priority (low | def | high) |
             Block }
```

Square brackets indicate optional elements. Curly brackets indicate lists (collection of values). Not all possible combinations of schedule/event/action/duration components result in valid policies. For instance the policy shown in Fig.2 does not present to the user a tile to set-up a duration because a "tweet" is an one-off instantaneous action. On the other hand, applying a prioritisation action for a given duration makes sense (Fig.1).

When enabled, a policy triggers actions and activates the required monitoring processes. If a "schedule" expression is

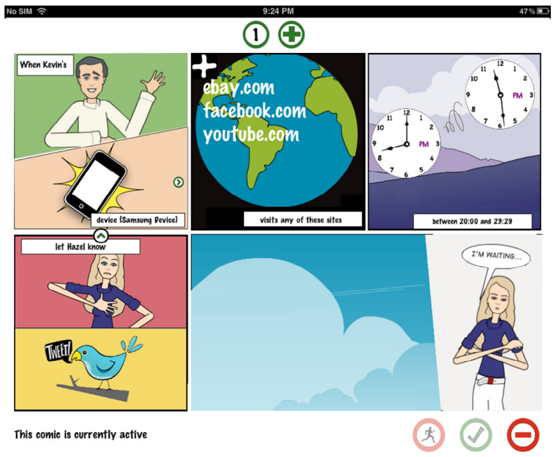

Fig. 2. The comic strip application running on an iPad

TABLE I
POLICIES FOR USE CASE

Policy	Description
P_1	When device H_3 of user U_S consumes more than 60% of home download allowance, then block H_3
P_2	During weekdays between 20:00-23:00, when device H_g is used, send an SMS to Dad's (U_D) mobile phone
P_3	During 20:00-23:00 on any day, when any device of user U_D (H_2 and H_{sp}) accesses webapp.workplace.com then give this device high priority for 1 hour
	P_1 Specification
Event:	Allowance(0.6, down), ipAdddr(H_3)
Action:	Block, ipAdddr(H_3)
	P_2 Specification
Schedule:	(20:00, 23:00, $\{Mon, Tue, Wed, Thu, Fri\}$)
Event:	Used, ipAdddr(H_g)
Action:	Notify (sms, "U_D, son plays online game")
	P_3 Specification
Schedule:	(19:00, 22:00, $\{*\}$)
Event:	Visits(webapp.workplace.com), ipAdddr(H_2, H_{sp})
Action:	Priority(high), ipAdddr(?)
Duration:	hours(1)

used, then, it defines time periods for certain weekdays, during which a policy (if enabled) can have an effect on the network's state by applying the specified actions. Outside this time period, actions cannot be applied but the associated monitoring processes keep running.

Home network management policies are actually an extension of the traditional obligation policies [23] and thus, they allow the specification of events which trigger a given set of related actions. On top of that, they also support schedule expressions and time-duration for actions, features which require extra state information and more sophisticated state management. We have implemented a small set of event types based on user input. "*Visits*" are events that are generated whenever a user accesses an Internet location (URL or IP address) using any device that he owns. "*Used*" events correspond to any use of a given device or set of devices, via user-related network activity. "*Allowance*" events correspond to the amount of uplink / downlink data that a user has consumed during some period using one or more devices. The volume of data is expressed as a percentage of the "home's overall allowance" with their Internet Service Provider.

The third component of a policy specifies one or more actions. If no events are specified, the actions are applied the time the policy is enabled, and get revoked when the policy is disabled or removed. Alternatively, if an event expression is used, actions are triggered dynamically at runtime. Currently we have implemented three types of actions. A user may be notified via a specific service (e.g. SMS, Email) when particular network usage criteria are met (depending on the event definition). Prioritisation actions are used to assign priorities to network resource usage among users / devices. There are three classes of priorities, normal, which is the default for all traffic, high and low. Prioritisation has been used by some users as a mean to penalise misbehaving individuals, banning indirectly certain applications (e.g. no live video streaming) and also to improve the quality of experience for users who actually need it for important tasks. The third type of action is to totally block someone from accessing the Internet. Our system supports even more types of actions (e.g. bandwidth allocation, fine-grained service access control) but we won't make further reference to them in the present work.

In general the applied actions are automatically revoked when the respective policies are disabled or removed by the users or when they exit their scheduled date-times. The fourth component of our policies indicates a duration over which an action will have effect. For instance, one may wish to block a device from accessing the Internet for a given amount of minutes or hours. When this time duration expires the actions are revoked (if revocable) and the policy is ready to fire again. "Hidden" events also exist that are automatically generated by our runtime environment and can trigger the revocation of an action: there is a configurable monthly roll-over event which resets all data-volume usage counters and forces the related allowance-based policies to revoke their actions.

D. Use Case

Here we introduce a typical scenario of a home network that will be used to elaborate our approach further. We assume that the network consists of a gaming console (H_g), two laptop computers (H_1, H_2), a desktop PC (H_3) and one smartphone (H_{sp}). There are three users, dad (U_D), mom (U_M) and son (U_S). Ownership of devices is as follows: H_3 belongs to U_S, H_1 belongs to U_M, H_2 and H_{sp} belong to U_D, and the gaming console (H_g) belongs to all family members (home). We also assume that the home network runs the policies which are shown in TABLE I. Most of the policy components are optional. Even short policies like P_1 do not have straightforward execution semantics: once P_1 "fires" its actions can be revoked either by disabling/removing it or by receiving a monthly data volume counter reset event. Our gateway runs a tweaked DHCP server and device to IP address mappings do not change over time to avoid inconsistencies.

E. Policy Engine: Managing home network policies

The "policy engine" (see Fig. 1) accepts new requests from the iPad user interface (UI), manages all running policies, and maintains the system overall state. The interaction between the UI and the engine is bidirectional. New requests arrive from the UI but individual policies and the network's overall state are also communicated back to it, so that users can have an up-to-date view of their home network. The engine uses persistent storage to log changes and also to keep snapshots of the current running configuration, which are used to recover the overall system state after a reboot (e.g. after a power-cycle).

When handling a new policy, a sequence of steps are executed by the policy engine. First, the supplied specification is checked for syntax errors and invalid values (e.g. malformed format of URLs and addresses, use of non local addresses in subject expressions, wrong time format, duplicate or non-existing policy IDs). The policy is then converted into a managed object that provides an interface that supports the following methods: enable / disable, schedule, obtain state information, send events and destroy. The engine uses this interface to manage the running policies. More details regarding the event handling and the enforcement and revocation of actions are provided in the following two subsections.

F. Policies and event processing

Here we explain the execution model of a policy object, with respect to incoming messages from other components, and also with respect to scheduling. In general, there are a variety of messages that may be received: create, enable, disable, destroy, date time period start-stop, and action timer expiration. We differentiate these messages which are used for the self-management of policies from the actual network usage events which they subscribe and trigger the execution of actions. Policies are only attached to the system's event bus (and thus can receive events) when they are both enabled and being within the date-time period (if specified). However, not every policy has the same behavior because they optionally contain a day-time expression, an action duration and a network usage event. If no usage event is specified, execution is quite straightforward and thus omitted from the paper. Policies P_1, P_2 and P_3 all have an event definition, which means that actions are triggered dynamically upon reception of the subscribed events.

In Fig. 3 we illustrate an execution scenario for our use case. The X-axis represents time, and during the shown time-window a series of events occur. For sake of simplicity we assume that Events A, B and C are all of the same type (e.g. H_2 or H_{sp} visit webapp.workplace.com) and all policies (P_1, P_2, P_3) subscribe for this event. On t_1 all policies are enabled and on t_8 they are disabled. While P_1 is being enabled (indicated with a green colour), *event A* triggers the invocation of its action, so host H_3 stays blocked from t_3 until t_8. Events

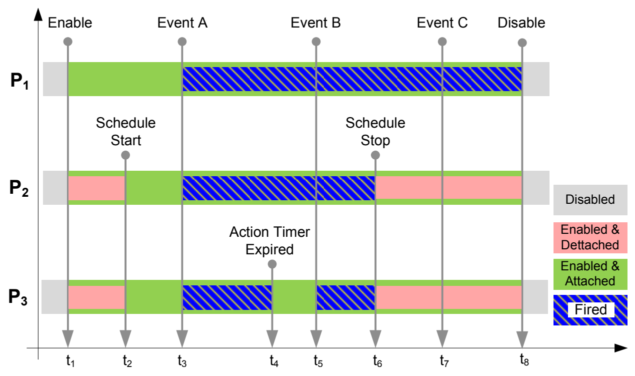

Fig. 3. An execution scenario for the case study policies

B and C have no effect because actions have been applied already from *event A*. At t_8 blocking of H_3 is revoked.

Policy P_2 contains a day-time schedule, which means that despite being enabled between t_1 and t_8, it won't accept incoming events outside the t_2 - t_6 period as it is detached from the event bus. This is why actions only fire from *event A* and not also from *event C*. There is no need for revocation, since "notifications" cannot be rolled back.

Policy P_3 is the most complex example, because it does have scheduling, event and action duration all together, and its action is revocable (prioritisation). Similarly to P_2 it won't trigger (fire) prioritisation outside the scheduling period (t_2 - t_6), despite being enabled. What is different, however, is that actions are only applied for 1 hour, thus, on t_4 priorities are set back to their default. Then, *event B* forces P_3 to fire its action again, until t_6 when the time duration expires.

G. Runtime Environment

Here we provide insights about the lower level details of network monitoring and configuration. Network usage events arrive via an event bus using a publish-subscribe model, and blocking, prioritisation and notifications are applied instantly when a policy "fires" its action(s). The inherent complexities of the aforementioned network management tasks are hidden from the policy engine by the underlying runtime environment which is shown in Fig. 1. This environment consists of two main subcomponents which provide to the above layers the necessary abstractions and allows policies to use a syntax with high-level concepts, decoupling them from the low-level details of network devices and protocols.

The configuration service (Fig. 1) allows the home networking infrastructure to be viewed, verified and reconfigured at a higher level of abstraction [22]. The overall configuration of the network is represented as a collection of platform-independent task descriptions, which specify the desired behaviour for certain network entities. Tasks define both the default and entity-specific behaviours. The defaults have effect when no other settings exist for an entity. Our prototype currently supports three types of tasks: host blacklisting (disconnect from the network), IP-layer access control and bandwidth allocation. Each type of task description has its own syntax and semantics, however, we support a set of common abstractions (subjects, targets, roles, groups, constraint expressions) to facilitate cross-task and global configuration reasoning. The reader is referred to [22] for more details.

The home network policies manifest themselves into an extension of the traditional obligation policies. This is more obvious when we consider instances like the policies of our case scenario, which include an event definition in their specification. Events trigger actions, and thus, event generation plays a central role in our design. At the device and network flow monitoring layer, we use the "Homework Information Plane" (HwIP) ([20]) architecture. At the heart of the HwIP resides the Homework Database (HWDB) which consists of an ephemeral and a persistent component. The ephemeral component is a high-speed custom in-memory database that support continuous (SQL) queries. The persistent component periodically saves the epheremal state in a relational database.

A wide array of network state and usage information (active network flows, http requests, wireless signal properties, dhcp leases etc.) is stored in the ephemeral database. Data is organised into tables as a time series (timestamp is used as the primary key) and memory management is done in a round-robin manner i.e. the oldest data will be dropped to make space for new data. Entities which wish to retrieve data or even subscribe for certain events can use continuous queries to describe simple or more advanced data patterns. Long running queries can run in the background and generate events whenever certain condition criteria are met. As new data arrive, associated queries are evaluated on the fly and if an event is generated it is communicated back to its subscribed entities.

H. An end-to-end example

In this section we consider how P_3 (TABLE I) is enforced. User U_D (Dad) builds a comic strip story in the UI by selecting, ordering and configuring tiles with appropriate values. His goal is to make sure that whenever he accesses his work network from home, he gets enough bandwidth resources so that Internet based applications operate smoothly.

When Dad submits the policy, the UI generates a textual representation of the policy and forwards it to the engine running on the Home Router. The engine analyses the request, checks for errors and inconsistencies and based on the individual components, instantiates and enables a policy object for it. Since policy P_3 has a day-time schedule and also an action duration, the appropriate time controls are embedded in the new object, which is self-managed. Prioritisation for device H_2 or H_{sp} is triggered dynamically from events of type "visits". The engine installs a new subscription for this event to Homework database (HWDB) via the monitoring manager.

Whenever the HWDB inserts a tuple in the *urls* table that matches "webapp.workplace.com", it generates a new event and forwards it to the engine's event bus. Policy P_3 being attached to it receives the event. Prioritisation actions are then triggered based on the information that the event carries about the device which accessed the work location. The policy forwards a request to the configuration service which then

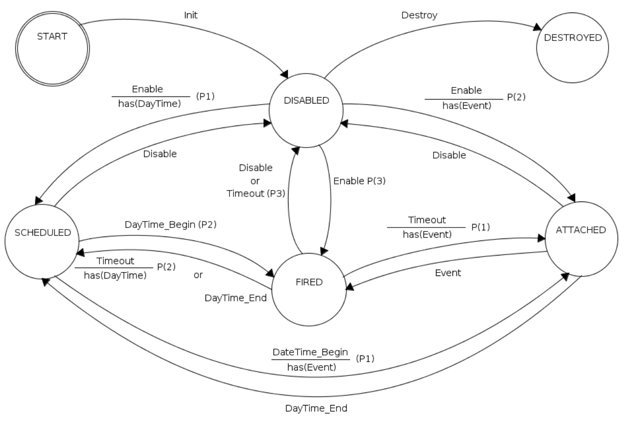

Fig. 4. User Policy State Machine

assigns the device of interest into the "high priority" class. Revocation of actions is done via the configuration service occurs based on the principles discussed in section II-F. Error handling mechanisms are deployed across the pipeline of execution and present meaningful error messages back to the user via a chain of cascading callbacks.

III. Implementation

Our home router is a Linux-based netbook running Ubuntu Server, OpenVswitch and Nox ([20], [24], [25] [26]). The system architecture presented in the previous section (Fig.1) mirrors also the organisation of our prototype. The details of each software component is further described in the remainder of this section.

The user interface (UI) is developed for the iPad (Fig.2). Its appeal allowed us lure users to (playfully) experiment with writting policies throughout a household over extended periods of time. The interaction of the UI with the policy engine was done by the UI inserting policy tuples into the ephemeral database and the policy engine subscribing to them. A custom NOX controller ([25], [26]) implemented replacement protocols for DHCP and DNS forwarding, and allows for an easier experimentation with policies related to network access control. The custom DHCP implementation also guarantees that devices with a given MAC address, always obtain the same IP address.

The policy engine is implemented as a library extension of the Ponder2 system [23]. Ponder2 is a distributed object management system which includes class frameworks for hierarchical domains (directories), policies over domains, a distributed event service, and a scripting language based on Smalltalk (PonderTalk). We have extended the existing obligation policies of Ponder2 in order to implement the home network policies, supporting the features discussed in the previous section (schedule, action duration). This design allows policies to be used in a distributed manner, making it possible to enforce and manage them as local objects, even when they reside on remote hosts. This feature would allow the home network's management to be further managed by the ISP or remote third parties, for example cloud-based providers.

Our management objects handle their state autonomously. They accept command strobes from the policy engine module and receive events from the proximity event bus of Ponder2. The implementation of policies was challenging due to the big number and the diversity of possible instances resulting from the combination of their partial components. For this reason, we implemented them as finite state machines (FSMs), using the open source Tungsten FSM framework. Whenever a new policy is created, a new FSM is also instantiated automatically and is integrated inside the policy object. Regardless of the specification of the policy, the state machine has always the same number of states, and it handles the differentiated behaviors among the various policies by using predicates in the transition rules. For example, a transition to the attached state only occurs if the policy specification does actually have an event component (*has(Event)* evaluates to true).

In Fig.4 we illustrate a policy's state machine which emulates the runtime behaviour discussed in section II-F. There are six states in total but *Scheduled*, *Fired* and *Attached*, can all be grouped under a global *Enabled* state. Transitions are labelled with the events which trigger them, and some of them have also conditions (e.g. has(DayTime), has(Event)). There are some cases where the same event type is used from more than one transitions outgoing from a given state. To resolve this issue we used priorities (e.g. P(1), P(2)) between ambiguous transitions. For example consider policy P_3 and assume that its FSM is in *Fired* state. When an action timeout event occurs (action duration of 1 hour), there are actually three candidate transitions. Policy P_3 has both an event and a daytime expression in its specification. Transition to state *Attached* is selected because it has a higher priority ($P(1) > P(2)$).

The actions which are currently supported in our prototype are blocking, prioritisation and notifications. Notifications are handled by a separate service which provides an extensible set of services like SMS, Email, Growl, Facebook etc. The notification service requires a few configuration steps to install user profiles for each service type. Blocking is handled via a NOX module which can blacklist a device and prevent it from accessing the home network. Priorities are implemented by the home network configuration service [22].

We have extended the configuration service to support priorities also, but not in strict networking terms. More specifically, the Linux traffic shaping interpreter modules were reworked to comply with the tc hierarchy shown in Fig.5. There are no prio queues used because of the known bandwidth starvation issues when a user with high priority fully utilises the available network resources. Instead, we have emulated three priority classes by assigning a bandwidth share (*min*) and a ceil limit (*cap*) to each of the three classes. The assignment shown in Fig.5 are our custom setup for a home with a speed of 10Mbps for downlink. A similar tree is also used to shape the uplink resources. When a class does not use its assigned resources (*min*), other classes may borrow the unused bandwidth, which can never exceed, however, the *cap* limit. The allocations for each class are easy to change since the configuration service setup is parameterised, and shares can

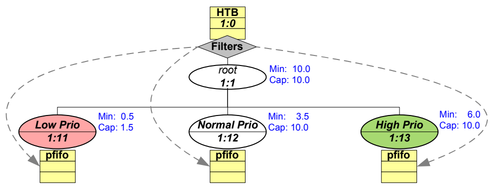

Fig. 5. Linux TC architecture

be expressed as percentages of the total available bandwidth. Filters classify traffic based on netfilter mark values which are set via the iptables interface. The tc HTB tree structure is formed once during initialisation, and during runtime, the actions of prioritisation policies only modify the set of iptable marking rules. Revoking a priority action is simply done by uninstalling the filter expression.

IV. Evaluation

In this section, we make a first effort to evaluate our work, by deploying our prototype in a UK household and performing a few experiments in order to confirm our system's behavior with respect to the running policies. We have actually installed two policies from our case scenario (section II-D): P_1 and P_3. We have also tested a number of other policies during this setup, but due to lack of space, we will only focus on the aforementioned two.

Our router box's traffic shaping modules were configured to 10 Mbps downlink and 1 Mbps uplink speeds, matching home's ADSL connection speed characteristics. For better control and for simplifying our measurement practices we used iperf to generate artificial traffic. However, all flows in our experiments connect to / from remote iperf hosts, so that we can give a more realistic flavour in our study. We had to enable port forwarding at the gateway in order to make it possible to accept remote iperf connections. All hosts involved in the measurement process were synchronized with NTP. In addition, we used switched Ethernet instead of WLAN, in order to avoid the performance fluctuations due to the varying quality of the links. Nevertheless, we have also conducted experiments over wifi, and obtained more "spiky" plots.

It is quite common that certain Internet service providers have plans where the total amount of downloads / uploads is limited over the month, and they also may throttle the connection of the home if excessive data are downloaded over the peak hours. Policy P_1 is well-suited for such cases, as it triggers the blocking of son's host (H_3) when it consumes more than 60% of home's monthly allowance which we have set to 22 GBytes. In Fig.6 there are illustrated two plots. The upper plot indicates the current data usage of host H_3 over time. The limit of $\sim$ 13.2 Gbytes ($22 * 0.6$) is indicated with the blue line, and it is reached roughly around second 120. The underlying plot shows the used datarate during the experiment of two *tcp iperf* download flows1. Until second 120 that no blocking is performed yet, they share almost equally the available bandwidth of 10 Mbps. Then, when the allowance limit is reached, host H_3 is blocked which explains the sharp

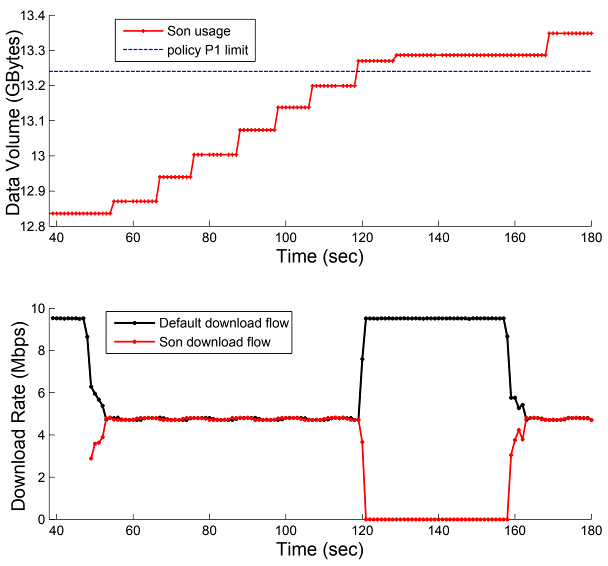

Fig. 6. Evaluation of policy P_1 blocking effects

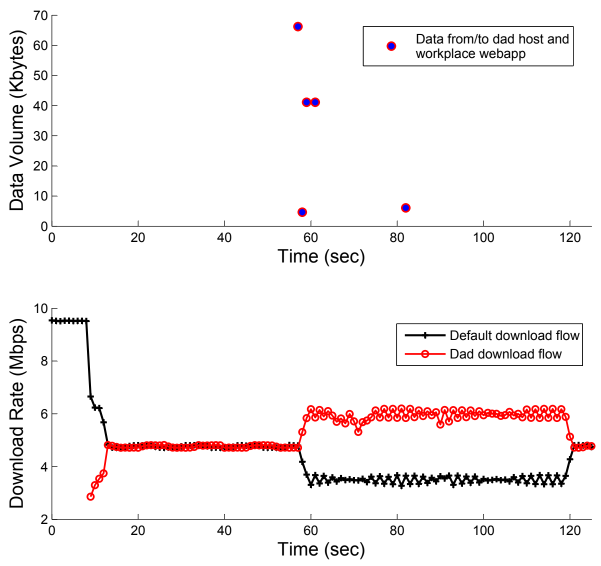

Fig. 7. Evaluation of policy P_2 traffic shaping effects

drop of the flow to son's host (red colour). The black flow (to mom's host, black colour) now monopolizes network resource usage until the point that policy P_1 is manually disabled and its actions are revoked ($\sim$second 158).

The case of policy P_2 is quite different from the previous scenario. Whenever one of the hosts of user U_D (H_2, H_{sp}) visits "webapp.workplace.com", this device gains high priority so that "dad" can effectively work from home using potentially demanding applications (e.g. remote desktop etc). In order to verify the effects of policy P_2, we have set up a monitoring module which counts all bytes (upload / download) from to location "webapp.workplace.com". These measurements are shown on the upper plot in Fig.7. Host H_2 exchanged roughly 70 Kbytes with the host of interest on second 57 for first time. This event should normally trigger a prioritisation action for H_2. The second sub-plot presents the download rates of two tcp flows, one towards son's host (default, black) , and the second one is towards dad's host H_2. Initially, both flows shared the available bandwidth on a fair basis, however, when H_2 accessed "webapp.workplace.com" (second $\sim$ 57), dad's host was assigned high priority and the respective flow, jumped to 6 Mbps (60% of the 10 Mpbs). Referring to the Linux tc model shown in Fig.5, this behavior is the result of redirecting the red flow into class 1:13. It worths mentioning here, that during deployment, users are informed that if many people/hosts are assigned with high priority, they may experience poorer performance than users in the default class, if they are considerably less.

V. Related Work

In the context of home networks, there hasn't been much work on developing new management paradigms. An exception is presented in [13] where authors make a first effort to change the way people handle and perform management tasks by pushing them to the cloud in the form of services. Their work outlines the requisite challenges, but is theoretical and is mainly focused on latency issues. HomeMaestro ([18]) is another example where bandwidth resources are shared among the users, introducing the notion of "application fairness" based on user's feedback and application-specific weights. This work too, however, is tightly coupled only to a specific aspect of management as it mainly deals with the allocation of resources given application-specific constraints. Management of faults in the configurations of home networks has also been recently studied ([27], [28]).

In enterprise networks, policy-based management has gained acceptance as a flexible approach for addressing many requirements for configuration, security, performance and fault-tolerance [29]. Use in home networks has been limited to simple policies for router configuration, wireless channel selection [15], bandwidth management [30], and traffic prioritisation [14]. A policy-based management system that supports a wider range of policies, better models and abstractions for home networks and their users is needed.

In ([31]), the authors propose a policy based architecture which supports multiple abstraction levels. In ([14]), the same authors demonstrate their policy based design to facilitate quality of service extensions and security management. Despite being promising, their work is mainly limited to high level description of architectures and potential implementation issues. Finally, in [32], the authors propose a dynamic resource allocation scheme, trying to optimise allocations across wireless and wired devices in a weighted fair manner. Similar to HomeMaestro applications may have different weights and based on their values the system performs host coordinated rate control to optimise network's overall performance.

We believe that our design complements the efforts of [13] in the sense that we provide the runtime tools to aid the transparent management of home network infrastructures. However, our main focus is to adopt a rather user-centric approach and towards this direction, this paper extends our previous work that abstract the low-level details and provide a transparent network configuration service ([21], [22]).

VI. Summary

This paper is motivated by the fact that typical home users do not possess the skills to manage effectively their networks as they keep becoming increasingly complex. Home networking infrastructures need to evolve and become easier to use and manage. To this end, we have presented a management framework for home networks which builds upon the abstraction of user-centric policies. Users create "comic strip" stories that relate to actual management tasks via an engaging and "fun to use" interface. Comic strips are then translated into an internal policy model which still uses a higher level notions for network entities, events, timing and actions. The described system has been implemented on a custom home network router, which is currently being deployed at several UK households. Our future work includes the extension of policies to support a richer set of management tasks, with focus on adaptive network resource management. We also plan to conduct user studies and evaluate the actual effectiveness of our approach, especially in terms of usability.

Acknowledgement

This research was supported by UK EPSRC research grant EP/F06446/1 (Homework).

References

[1] "Global broadband internet subscribers, q3, 2011." [Online]. Available: http://www.ispreview.co.uk/story/2012/01/13/global-broadband-internet-subscribers-total-581-million-in-q3-2011.html

[2] R. E. Grinter, W. K. Edwards, M. W. Newman, and N. Ducheneaut, "The work to make a home network work," in *Proceedings of the ninth conference on European Conference on Computer Supported Cooperative Work*. New York, NY, USA: Springer-Verlag New York, Inc., 2005, pp. 469–488.

[3] R. E. Grinter, W. K. Edwards, M. Chetty, E. S. Poole, J.-Y. Sung, J. Yang, A. Crabtree, P. Tolmie, T. Rodden, C. Greenhalgh, and S. Benford, "The ins and outs of home networking," *ACM Trans. Comput.-Hum. Interact.*, vol. 16, 2009.

[4] P. Brundell, A. Crabtree, R. Mortier, T. Rodden, P. Tennent, and P. Tolmie, "The network from above and below," in *Proceedings of the first ACM SIGCOMM workshop on Measurements up the stack*, ser. W-MUST '11. New York, NY, USA: ACM, 2011, pp. 1–6.

[5] P. Tolmie, A. Crabtree, S. Egglestone, J. Humble, C. Greenhalgh, and T. Rodden, "Digital plumbing: the mundane work of deploying ubicomp in the home," *Personal Ubiquitous Comput.*, vol. 14, pp. 181–196, 2010.

[6] W. K. Edwards, R. E. Grinter, R. Mahajan, and D. Wetherall, "Advancing the state of home networking," *Commun. ACM*, vol. 54, pp. 62–71, 2011.

[7] A. Akella, G. Judd, S. Seshan, and P. Steenkiste, "Self-management in chaotic wireless deployments," in *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2005, pp. 185–199.

[8] K. Papagiannaki, M. Yarvis, and W. S. Conner, "Experimental characterization of home wireless networks and design implications," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, April 2006, pp. 1–13.

[9] N. Hu, L. Li, Z. Mao, P. Steenkiste, and J. Wang, "A measurement study of internet bottlenecks," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, pp. 1689 – 1700 vol. 3, march 2005.

[10] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, "Characterizing residential broadband networks," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 43–56.

[11] X. Xing and S. Mishra, "Where is the tight link in a home wireless broadband environment?" in *MASCOTS '09*, 2009, pp. 1–10.

[12] S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè, "Broadband internet performance: a view from the gateway," *SIGCOMM Comput. Commun. Rev.*, vol. 41, pp. 134–145, Aug. 2011.

[13] C. Gkantsidis and H. Ballani, "Network management as a service," Microsoft Research, Technical Report MSR-TR-2010-83, 2010.

[14] A. I. Rana and M. O. Foghlú, "Policy-based network management in home area networks: interim test results," in *Proceedings of the 3rd international conference on New technologies, mobility and security*, ser. NTMS'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 334–336.

[15] D. Pediaditakis, L. Mostarda, C. Dong, and N. Dulay, "Policies for self tuning home networks," in *Proceedings of the 10th IEEE international conference on Policies for distributed systems and networks*, ser. POLICY'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 29–32.

[16] P. Bull and M. Harrison, "Managing broadband home networks," *BT Technology Journal*, vol. 24, pp. 79–85, 2006.

[17] J. Yang, W. K. Edwards, and D. Haslem, "Eden: supporting home network management through interactive visual tools," in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. New York, NY, USA: ACM, 2010, pp. 109–118.

[18] T. Karagiannis, E. Athanasopoulos, G. Christos, and P. Key, "Homemaestro: Order from chaos in home networks," Microsoft Research, Technical Report MSR-TR-2008-84, 2008.

[19] S. Zeadally and P. Kubher, "Internet access to heterogeneous home area network devices with an osgi&#45;based residential gateway," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 3, pp. 48–56, December 2008.

[20] J. Sventek, A. Koliousis, N. Dulay, D. Pediaditakis, T. Rodden, T. Lodge, O. Sharma, M. Sloman, B. Bedwell, K. Glover, and M. Richard, "An information plane architecture supporting home network management," in *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management*, 2011.

[21] D. Pediaditakis and N. Dulay, "Verifying home network bandwidth sharing plans," in *6th International Conference on Network and Service Management (CNSM)*, 2011.

[22] D. Pediaditakis, G. Anandha, N. Dulay, and M. Sloman, "A configuration service for home networks," in *Proceedings of the IFIP/IEEE Network Operations and Management Symposium, Maui, Hawaii, USA*, 2012.

[23] K. Twidle, N. Dulay, E. Lupu, and M. Sloman, "Ponder2: A policy system for autonomous pervasive environments," *International Conference on Autonomic and Autonomous Systems*, pp. 330–335, 2009.

[24] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 105–110, 2008.

[25] "A nox controllable home network router." [Online]. Available: https://github.com/homework/homework

[26] R. Mortier, B. Bedwell, K. Glover, T. Lodge, T. Rodden, C. Rotsos, A. W. Moore, A. Koliousis, and J. Sventek, "Supporting novel home network management interfaces with openflow and nox," in *Proceedings of the ACM SIGCOMM 2011 conference on SIGCOMM*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 464–465.

[27] C. Dong and N. Dulay, "Argumentation-based fault diagnosis for home networks," *ACM SIGCOMM Workshop on Home Network (Homenets 2011) , Toronto, Canada*, 2011.

[28] B. Aggarwal, R. Bhagwan, T. Das, S. Eswaran, V. N. Padmanabhan, and G. M. Voelker, "Netprints: diagnosing home network misconfigurations using shared knowledge," in *6th USENIX symposium on Networked systems design and implementation*, Berkeley, CA, USA, 2009.

[29] A. K. Bandara, N. Damianou, E. C. Lupu, and M. Sloman, "Policy based management," in *Handbook of Network and System Administration*, J. Bergstra and M. Burgess, Eds. Elsevier, 2008, pp. 507–564. [Online]. Available: http://oro.open.ac.uk/19438/

[30] S. Jha and M. Hassan, "Java implementation of policy-based bandwidth management," *Int. J. Netw. Manag.*, vol. 13, pp. 249–258, July 2003.

[31] A. Rana and M. Foghlu, "New role of policy-based management in home area networks - concepts, constraints and challenges," in *New Technologies, Mobility and Security (NTMS), 2009 3rd International Conference on*, 2009, pp. 1 –6.

[32] C. Gkantsidis, T. Karagiannis, P. Key, B. Radunovic, E. Raftopoulos, and D. Manjunath, "Traffic management and resource allocation in small wired/wireless networks," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. New York, NY, USA: ACM, 2009.