A Configuration Service for Home Networks

Dimosthenis Pediaditakis, Anandha Gopalan, Naranker Dulay and Morris Sloman
Department of Computing
Imperial College London
London, SW7 2AZ, United Kingdom
Email: {d.pediaditakis, a.gopalan, n.dulay, m.sloman}@imperial.ac.uk

***Abstract*—As the complexity and size of home networks increases, so does the need for better management support for non-expert home users. In this paper we describe an extensible policy-based configuration service that allows the underlying networking infrastructure to be viewed, verified and reconfigured at a higher-level of abstraction. The aim is ensure that configuration changes in network infrastructure do not result in problematic setups and to allow user-centric tools to operate at a higher-level of abstraction. Our implementation compiles configuration tasks to host/device level commands and supports deployment to multiple heterogeneous devices. We demonstrate and evaluate our approach, using access control and bandwidth sharing as examples of home management tasks.**

I. Introduction

It has been estimated that there were over 500 million broadband subscribers worldwide in 2010 [1]. This number is increasing as is the diversity of network-enabled devices and applications. Despite this growth, little work that has been done on making home networks easy to manage for home users who lack technical knowledge and often have their own very different understanding of networking concepts ([2]).

Studies have also highlighted other differences with enterprise networks. For example, homes suffer from arbitrary (chaotic) deployments [3] that lead to loss of performance and availability ([4], [5]). Upload and download utilizations are not highly asymmetric and bursty traffic is more intense than in enterprise environments ([6] and [7]). The speeds of broadband connections can vary dramatically over the course of a day due to poor physical connections, radio interference and ISP throttling [8]. This can be particularly annoying to home users who believe they were promised a certain broadband speed at all times. Also, the organisation authority in a home is casual.

Home users would certainly like their networks to "just work". However a hands-off/black-box approach where home users have no control of the network is not always desired. Ethnographic and empirical studies ([9], [2]) have suggested that among the most popular aspects of home networking that users wish to manage are access to web-sites and services and the regulation of network utilization (mostly bandwidth). This is particularly evident in families where a parent may wish to (i) limit a child's access to websites or the time they spend using the Internet, (ii) prioritize traffic, for example, that a VPN connection to the office is more important than a teenager's multi-user game, or a that a light/occasional user should have priority over a frequent/heavy-user.

In response to such issues, a number of recent papers have proposed new ideas for managing home networks ([10], [11], [12], [13], [14]) while a few only have developed prototype implementations. These proposals and implementations, lack however, a configuration service, that allows the underlying networking infrastructure to be viewed, verified and reconfigured intuitively. The aim of our work is to ensure that changes in home network's configurations do not result in problematic setups and to allow user-centric tools to operate on the network at a higher-level of abstraction. Our implementation compiles configuration tasks to host/device level commands and supports deployment to multiple heterogeneous devices. We demonstrate and evaluate our approach, using access control and bandwidth sharing as examples of home management tasks.

II. Configuration Service

Home users perceive a home network at many levels. Different users can have different perceptions and different expectations and needs. This is easy to see in how they name things, e.g. Alice (a specific person), Visitors (a role), Mum (a specific person and role), Facebook (name of website/company/service), Email (name of implicit service), Hotmail (named explicit service), Outlook (name of application), Printer (device), Canon (named device). Naming is an important concept in home networking and ideally there shouldn't be an imposed view of how users name things. Rather it should be possible to map names to underlying networking abstractions and entities. Similarly for management, users should be able to express management requirements, where possible, using their own vocabulary and have it mapped accordingly.

To some extent, the management of a home network can be categorized using the classical FCAPS (Fault, Configuration, Accounting, Performance, Security) model. However, the emphasis in home networks is very different and leads to different solutions. One of the biggest challenges, is for user-interface designers to produce interfaces that are intuitive and natural. For this, they need management systems to provide higher level views of the network, its capabilities, and its behaviour.

In this paper, we focus on configuration management and in particular, on providing a service that verifies and enforces configuration actions on the home network infrastructure. Such actions will typically be requested by a user via an intuitive user-interface or automatically triggered by management policies responding to events in the network.

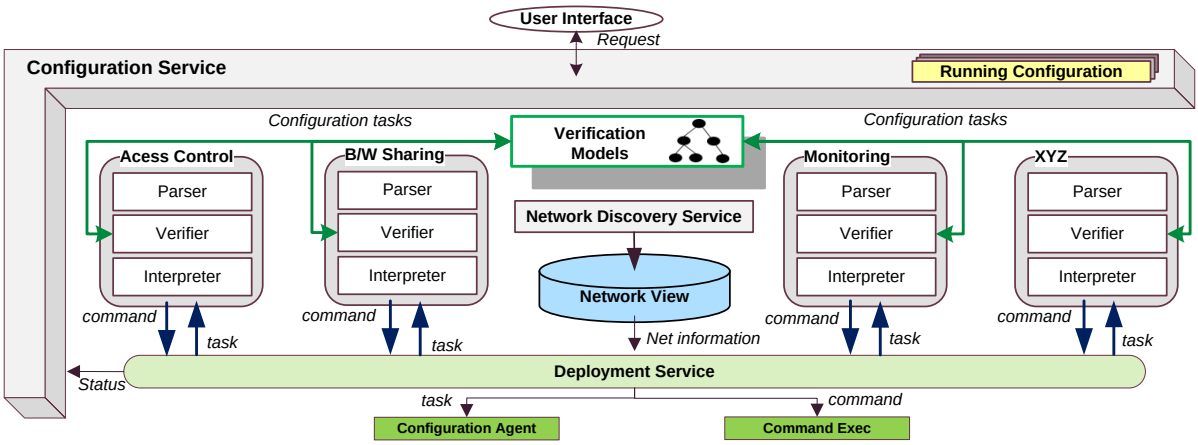

Fig. 1. Overall design of the configuration service

A. Design

In our design, the configuration of the home network (see Fig. 1) is represented as a collection of configuration task descriptions, which specify the desired behaviour for particular network entities. Tasks define both default and more specific entity behaviours. The defaults have effect when no specific settings exist for an entity. For example, consider a home network with a default policy which allows all users to access the home's network printer, and at the same time a more specific access control task, which specifically denies a visitor's laptop from accessing the printer. Although type of task description (access control, bandwidth sharing, accounting etc) can have its own syntax and semantics, our framework supports a set of common abstractions (subjects, targets, roles, groups, time, constraint expressions) to support cross-task and global configuration reasoning. A simple access control task description might be:

```
Tags:    {Son, Netbook}
Task:    {deny, outgoing}
Target:  {Host, sonNetbook}
Service:{Host, restrict.com, Proto,(www,ftp)}
```

This description used in the example is network-centric. In practice it should be geared to home users and use more user-centric concepts. It is a part of our future plans to work on this. Identifiers in this example can refer to entities or be functions that return a set of entities. In order to keep task descriptions concise we have avoided XML, even though it might be preferable in practice.

For each type of task description there is a *mini-compiler*, that is able to parse a description, verify it individually and in conjunction with the configuration of the current network, and generate device-specific commands for installation. The latter can be performed directly from the associated interpreter module for the given type of task, or alternatively, it can be distributed to target devices if they are able to host the interpreter. Device-level and traffic-level information about the network is gathered and kept updated in a "network-view" datastore using the Homework information plane [15].

The configuration of the current network can be named, saved and loaded on demand replacing the existing settings. For example, a family may have a "Weekend" configuration allowing online gaming, and a "Weekday" one which limits it. Saved configurations can also be applied dynamically by management policies in order to adapt the network.

B. Verifying home network configurations

We would like the overall behaviour of the network to be consistent and not lead to problematic networks. However, this is only possible if individual configuration tasks do not contain invalid parameters, do not conflict with each other, and do not overwhelm the network's overall resources. For example, if a task limits the download rate for a host to 3 Mbps, and at the same time another task assigns it a rate of 5 Mbps, this can potentially result in 2 Mbps that are wasted. Problems can also arise when a task conflicts with a more specific one. For instance when a subnet is blocked from accessing the Internet, we may need to unblock a specific host in it. It is thus important to have a mechanism which can help avoid configuration inconsistencies or at least resolve them possibly make the user aware of the negative effects of reconfigurations.

Referring back to our service's design (Fig.1), each type of configuration task implements its own verifier module. Verifiers employ custom algorithms and models, tailored to the type of configuration descriptions they handle. Models are shared among all verifiers to support more sophisticated reasoning. For example, it would be a waste of resources to block downloading traffic to a host and at the same time exclusively assign to it a share of the network's bandwidth resources. A detailed description of our verification procedure for bandwidth sharing is given in [16]. The procedure uses a tree-based structure to model and verify the network's bandwidth sharing plan, as new tasks are activated.

When a conflict is detected, the decision on what further actions to take depends on the system's conflict-resolution policies, which are implemented inside a "conflict resolver" module. Currently, we return an error and leave the requesting process to resolve it.

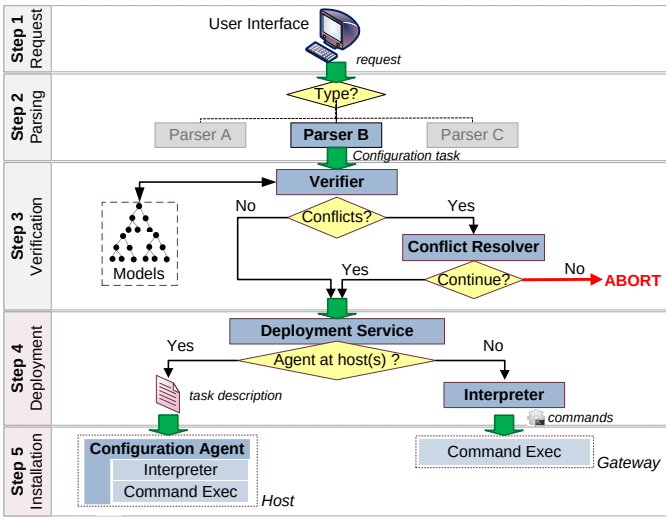

Fig. 2. Pipeline of applying a configuration task

C. Applying a configuration task

The configuration service handles reconfiguration requests one-at-time and step-by-step (see Fig.2). New task descriptions are issued via a user-interface or alternatively by management policies responding to events in the network. The type of the task is determined from the header (start of description) and the associated parser invoked. This parses the task into an internal representation and invokes the verifier which uses the current configuration to check for both task-level and global errors. If verification is successful, tasks are translated into system-specific commands and scripts and distributed to appropriate hosts. This is coordinated by the deployment service. Both translation and distribution require details of the running network which is maintained in "network view" datastore discussed in section II. Translation can be done either on configuration service host or on local devices (via a configuration agent). Configuration changes can result in changes on the main gateway or on local devices. The latter can be useful in some circumstances, for example to block an Internet service on a local device rather than on the gateway.

III. Implementation

A. Prototype Design

Our configuration service prototype is written in Java and we currently have interpreter modules for Linux-based router-boxes and Linux-based client machines. Ponder2 [17] is used as the basis for our application programming interface (API). Ponder2 is a distributed object management system which includes class frameworks for hierarchical domains (directories), policies over domains, a distributed event service, plus a scripting language based on Smalltalk (PonderTalk). Management policies can be defined using event-condition-action rules and as teleo-reactive processes. The configuration service is a part of the Homework project ([15]) that aims to provide a new user-centric approach to home networks.

B. Supported configuration tasks

The configuration service can receive both task descriptions and PonderTalk scripts. The latter provides more direct control of the service. Task descriptions are the basic unit used to create configurations. We currently implement three types of tasks: network access blacklisting (block at the link layer), access control (for already associated hosts) and rules for regulating the usage of bandwidth among users. We describe each in turn:

1) Blacklisting hosts: The first type of configuration task is used to prevent someone from connecting to the home network. For example, it may be desired to set a default policy which allows all devices to connect to the network except from a set of "black-listed" ones (or vice versa). Applying such a configuration task disallows the association with the home network's access point for the blacklisted devices, and also denies them a DHCP lease, even if they connect via wired Ethernet. The custom router box we've used is NOX ([18], [19]) enabled and blacklisting is implemented by sending a POST *http* request to a custom NOX controller. An example follows:

```
blacklistTask := add_blacklistHwAddress
 addr: ("00:11:6c:11:09:89:00:13:ff:10:09:11")
 desc: "Block Xbox and visitor's tablet".
```

2) IP-layer access control: This type of task manipulates the firewall setup of the home network. The interpreter module for the task generates Linux *iptables* commands, which are also compatible with the OpenWrt zones-based chain schema. During interpretation, the provided addresses/host-names/aliases are converted to network addresses and then identified either as a gateway host, a local client, an external address or as a special value. The special values "Internet" and "Local" are used to set the network's default policies.

```
accessTask := add_AcRule
 list: #( "allow" "incoming"
          #("dadDesktop" "dadLaptop")
          #("www.dad-work-location.com")
          #("SSH")  ).
```

The rule above requires two configuration settings to be applied instead of one. First, it installs port forwarding for tcp:22 by manipulating the "*Nat*" table of Netfilter. Second, it installs an extra rule in the "*Filter*" table, under the FORWARD chain (at the appropriate sub-zone) to allow the incoming traffic to reach port 22 of dad's laptop.

3) Bandwidth Allocation: Currently we support the following three types of bandwidth allocation tasks:

- Maximum rate: This is the maximum amount of bandwidth that a local host may use under any circumstance.
- Minimum assured rate: This is the minimum amount of bandwidth that will be available anytime for a local host. If the network resources are not used by the host, other devices may "borrow" them.
- Exclusively assigned rate: This statically assigns the specified rate to a local host. Unlike the minimum assured rate, other hosts cannot "borrow" the unused resources,

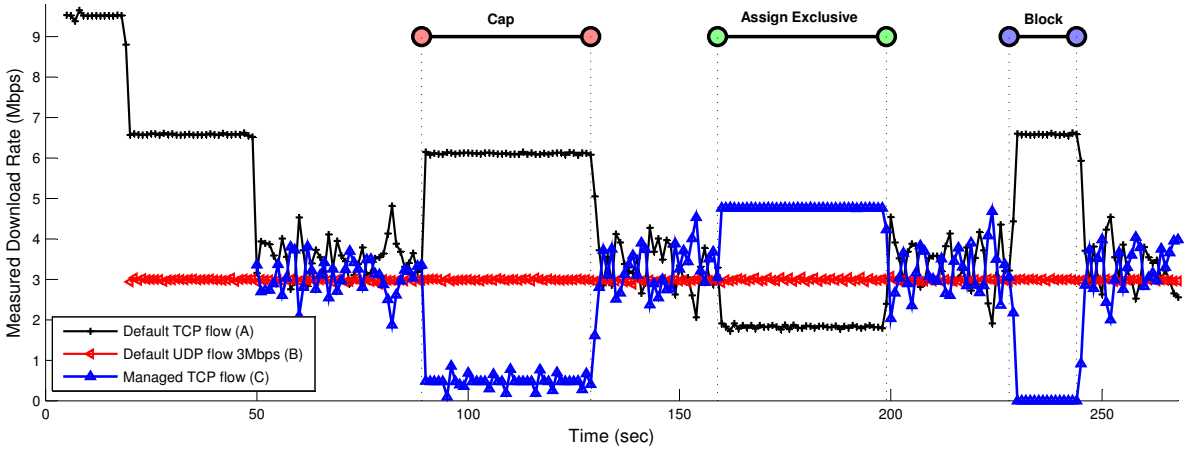

Fig. 3. Effects of applying a new configuration task

and thus delays are considerably smaller which in turn provides better QoS support.

A maximum cap rate example for Internet downloads is:

```
bwMaxRateTask := add_TcRule
 list: #( "Cap" "Download" 0.5 "Mbps"
       #("sonLaptop") #("Internet")  ).
```

For bandwidth tasks we generate Linux "*tc*" commands and use the hierarchical token bucket (HTB) queuing discipline (*qdisc*). Filters redirect outgoing traffic to the appropriate queues, and HTB classes control the usage of bandwidth resources from these queues in a hierarchical manner. Unfortunately, the HTB has certain shortcomings that required us to create *qdisc* hierarchies which under certain circumstances could be slightly different from the desired ones (see [16]).

C. Checking for inconsistencies

Incoming requests are firstly checked for syntax errors, based on the expected format of the task type. For instance, supplied values of hardware address, IP addresses, hostnames, special keywords and so on, are all validated during parsing. Individual task descriptions must also be sensible given home network's status, e.g. specifying an assured rate of 30 Mbps when home's downlink speed is 10 Mbps is erroneous and nonsense. Another useful functionality is to identify tasks with equivalent or opposite semantics, regardless of their exact syntax. However, the most interesting consistency checks are the ones which evaluate the network's overall configuration. This is achieved by employing custom verification algorithms, implemented inside the verifier for each class of configuration. In [16] we presented an algorithm for detecting conflicts in bandwidth sharing plans. The same model can be used to make checks for access control tasks, indirectly, by setting zero cap limits to the specified services.

Verifying configurations and applying them are closely related procedures, but nevertheless, in our design they are totally decoupled, giving a great level of flexibility when extending our service. Also, the models can be accessed from any verifier, promoting the cross-checking between different classes of tasks, whenever it is sensible. For example, blacklisting a device and the same time explicitly allowing others to access a backup service on it, is a mild conflict situation which, at least, might be of interest for the user.

IV. Experimental evaluation

A. From tasks to network behaviors

In order to evaluate our implementation, we have decided to test whether the network behaviour of a home network accurately reflects the semantics of its configuration. Our evaluation setup aims to emulate a typical home network. Our home gateway [19], is a netbook (EeePC S101) converted to a router-box running Ubuntu server along with custom software. We used switched Ethernet in order to avoid the performance fluctuations induced by wireless links. We attached two client machines (hosts A and B) on a switch connected to the router, each running both a *tcp* and a *udp iperf* server (ports 5001 and 5002). On host A, we ran an extra *tcp iperf* server (port 5003), where we direct the "managed" traffic. We used two extra netbooks, connected on the department's intranet, to generate artificial traffic directed towards the clients, inside the home network. The gateway's bridge and Internet interfaces were setup to a ceiling rate of 10 Mbps for downloads and 1 Mbps for uploads respectively, emulating the available bandwidth of a typical UK broadband Internet connection.

Our test case scenario (Fig. 3) demonstrates the effects of a variety of reconfiguration actions. At seconds 5 and 20 two flows start, *tcp* flow A connects to port 5001 of host A and *udp* flow B (at 3 Mbps) sends data to port 5002 of host B. We refer to flows A and B as "default" traffic because they are not affected by any configuration task. From seconds 20 to 50 we observe a sharp drop for flow A ($\sim$ 3 Mbps fall), because *tcp* adapts to the lower available bandwidth. At second 50, *tcp* flow C starts, and connects to port 5003 of host A. Flow C shares on a fair basis the remaining bandwidth ($\sim 9.7 - 3 = 6.7$ Mbps) with flow A. The oscillations observed in flows A and C are quite characteristic for combined *tcp* traffic which is served by a single *pfifo* queue.

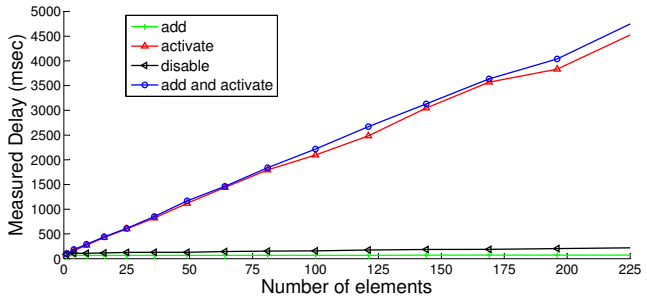

Fig. 4. Comparing the delays for add, apply, add-apply and disable

Between seconds 90 and 130 (Fig. 3), we limit the download rate for traffic destined to host's A port 5003. Flow C then drops to 0.5 Mbps and flow A "jumps" to roughly 6 Mbps, using the extra unsused bandwidth. Flows A and C have the same destination (Host A) but they get a totally different treatment. Before second 90 all flows were served from the same queue in gateway's bridge interface, under a common rate. After the reconfiguration, flow B is directed to a separate queue, with a lower service rate (0.5 Mbps).

Between seconds 160 and 200, we assign an exclusive rate of 5 Mbps to flow C and the effects of that change are clear in Fig. 3. The extra 1.5 Mbps that flow C gets (totaling to $\sim$ 5 Mpbs) are deprived of flow A. *Udp* traffic is greedy but this is not the case for the default *tcp* traffic (flow A) which is cross-traffic friendly. Having dedicated queues for exclusive rate assignements and non-shared transmission tokens minimizes the oscillations of *tcp* flows (Fig. 3).

The final configuration task blocks flow C and takes place between seconds 235 and 245. Flow C's rate drops to zero, and at the same time, flow A consumes the unused available bandwidth, climbing to $\sim$ 6.7 Mpbs. The effect of blocking a flow seems to have a slightly higher delay, compared to the traffic shaping actions (see the slopes of the download rate curves). This is because we use stateful firewall settings on the main gateway, and since *iperf* flows remain active throughout the whole duration of the experiment, we automatically flush the flow tables of the *iptables* connection tracking module, to avoid "established connection" treatment.

B. Re-configuration processing time

In this section, we aimed to assess the performance of our service, in terms of reconfiguration delays. It is of major importance to know "*how fast can the service apply incoming reconfiguration requests*". The answer to this question can help identify service's suitability for management applications that perform reconfiguration actions with a given frequency and have specific delay and scalability requirements. Knowing the performance characterisitcs and bottlenecks, it might be required to regulate correspondingly the rate of the incoming requests so that our system cope with the load.

Currently, our deployments are focused on single-host installations and thus, we do not include in our measurements the communication and DNS resolution delays. A distributed

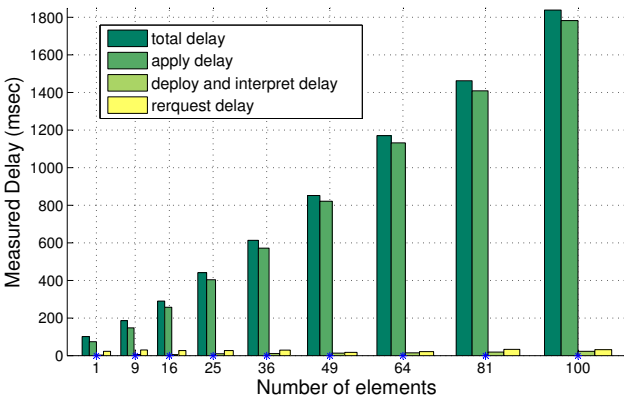

Fig. 5. Delay for adding-activating a configuration

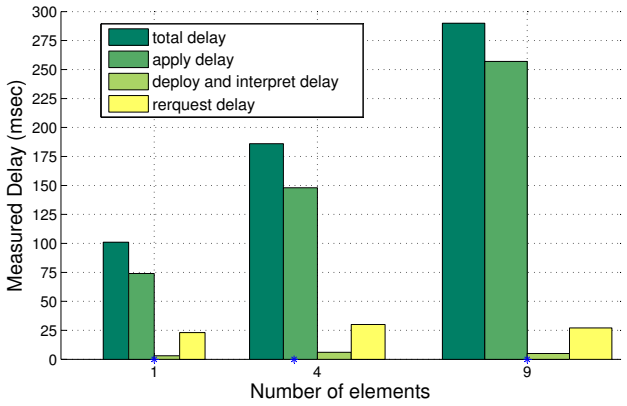

Fig. 6. Delay for adding-activating a configuration (1 to 9 elements)

setup, on the other hand, would increase the level of flexibility at the cost of performance. The client which initiates the request communicates over a local *tcp* socket interface with the configuration service. The service' forward the requests for verification, and then forwards it to the appropriate configuration agent via message passing. Finally, the agent interprets the description and generates commands that make system calls using the command line interface of iptables.

Figure 4 shows a comparison of the total delay times required for adding, applying, disabling and combined adding-applying an IP-layer access control task. Each point is the average delay measured over ten runs for a configuration of given complexity. The more elements (target and service endpoint addresses, source/destination ports, protocols) a task definition has, the more complex it is. It is obvious from Fig. 4 that the most intensive tasks are the ones which involve task activation, and also that the delay grows linearly with the number of a task's elements. Figure 5 gives a more detailed analysis of how much each delay component contributes in total, for the add-activate operation. Clearly, the apply delay monopolizes the total delay time. This indicates that the invocation of the iptables command line interface is the performance bottleneck. Nevertheless, for firewall reconfiguration tasks with less than 25 elements, the total delay is less than half a second. Figure 6 illustrates a close-up for the 1 to 9 elements cases, showing

that the delays induced by our prototype service amounts to less than 25% of the total, a percentage which drops as the complexity increases. In our measurements we haven't included the processing delay component because it was always nearly negligible ($< 1msec$).

V. Related Work

The related work in home network management has been limited to developing specific mechanisms or offering high-level architectures. In ([20]), the authors propose a policy based architecture which supports multiple abstraction levels. In ([11]), the same authors demonstrate how their architecture can facilitate quality of service extensions and security management. Despite being promising, the work is limited to a high level description of architectures and potential implementation issues. In HomeMaestro ([14]) bandwidth resources are shared among users, using the notion of "application fairness" based on user feedback and application-specific weights. In [21], a dynamic resource allocation scheme is proposed, that tries to optimize allocations across wireless and wired devices in a weighted fair manner. A more radical approach is suggested in [10] where management tasks are pushed to the cloud. The paper outlines a number of challenges but is mainly focused on latency issues. A cloud-based approach is also advocated in [22] which allows a home net to be viewed and managed as a collection of virtual subnets. We believe that our design complements the efforts of cloud-based solutions and has distinct advantages for monitoring and controlling local communications and for diagnosing faults in the home ([23]). Finally, policy-based management in home networks has been limited to simple policies for router configuration, wireless channel selection [12], bandwidth management [24], and traffic prioritisation [11].

VI. Conclusion

As the size, diversity and complexity of home networks continues to grow, so does the need for new user-centric management approaches. In this paper we have presented a configuration service, which facilitates the development of new management services for home networks. The service is extensible and aims to support a wider range of policies, better models and better abstractions. Network configurations consist of a set of conflict-free tasks that reflect the current state of the network at a higher level view. The aim is that changes to the configuration and changes to underlying network lead to the desired behaviour. Our implementation compiles tasks to device-level commands and supports deployment to multiple heterogeneous hosts. We have experimentally evaluated our prototype in terms of resulting network behaviour, and we have also characterized its delay properties. Our future work is focused on developing configuration specifications that are closer to the expectations and perceptions of users and and extending the range of configuration tasks available to management services.

Acknowledgement

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

References

[1] http://www.ispreview.co.uk/story/2011/03/24/world-broadband-internet-subscribers-topped-523-million-in-2010.html, 2010.

[2] M. Chetty, R. Banks, R. Harper, T. Regan, A. Sellen, C. Gkantsidis, T. Karagiannis, and P. Key, "Who's hogging the bandwidth?: The consequences of revealing the invisible in the home," 2010.

[3] A. Akella, G. Judd, S. Seshan, and P. Steenkiste, "Self-management in chaotic wireless deployments," in *11th annual international conference on Mobile computing and networking*. New York, USA: ACM, 2005.

[4] K. Papagiannaki, M. Yarvis, and W. S. Conner, "Experimental characterization of home wireless networks and design implications," in *INFOCOM 2006*, April 2006, pp. 1–13.

[5] D. Eckhardt and P. Steenkiste, "Measurement and analysis of the error characteristics of an in-building wireless network," in *SIGCOMM '96*. New York, NY, USA: ACM, 1996, pp. 243–254.

[6] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *9th ACM SIGCOMM*, 2009.

[7] K. Cho, K. Fukuda, H. Esaki, and A. Kato, "The impact and implications of the growth in residential user-to-user traffic," *6th ACM SIGCOMM*, 2006.

[8] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, "Characterizing residential broadband networks," in *7th ACM SIGCOMM*, 2007.

[9] R. E. Grinter, W. K. Edwards, and M. Chetty, "The ins and outs of home networking: The case for useful and usable domestic networking," *ACM Trans. Computer Human Interaction*, vol. 16, 2009.

[10] C. Gkantsidis and H. Ballani, "Network management as a service," Microsoft, Technical Report, 2010.

[11] A. I. Rana and M. O. Foghlú, "Policy-based network management in home area networks: interim test results," in *3rd IFIP NTMS*, 2009.

[12] D. Pediaditakis, L. Mostarda, C. Dong, and N. Dulay, "Policies for self tuning home networks," in *10th IEEE POLICY*, 2009.

[13] P. Bull and M. Harrison, "Managing broadband home networks," *BT Technology Journal*, 2006.

[14] T. Karagiannis, E. Athanasopoulos, C. Gkantsidis, and P. Key, "Homemaestro: Order from chaos in home networks," Microsoft, Technical Report, 2008.

[15] J. Sventek, A. Koliousis, N. Dulay, D. Pediaditakis, T. Rodden, T. Lodge, O. Sharma, M. Sloman, B. Bedwell, K. Glover, and M. R., "An information plane architecture supporting home network management," in *International Symposium on Integrated Network Management*, 2011.

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

[17] K. Twidle, N. Dulay, E. Lupu, and M. Sloman, "Ponder2: A policy system for autonomous pervasive environments," *5th International Conference on Autonomic and Autonomous Systems*, 2009.

[18] 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.

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

[20] A. Rana and M. Foghlu, "New role of policy-based management in home area networks - concepts, constraints and challenges," in *International Conference on New Technologies, Mobility and Security*, 2009.

[21] C. Gkantsidis, T. Karagiannis, P. Key, B. Radunovic, E. Raftopoulos, and D. Manjunath, "Traffic management and resource allocation in small wired/wireless networks," in *International Conference on Emerging networking experiments and technologies*, ser. CoNEXT, New York, USA, 2009, pp. 265–276.

[22] Y. Yiakoumis, K. Yap, S. Katti, G. Parulkar, and M. N., "Slicing home networks," *ACM SIGCOMM Workshop on Home Network (Homenets) , Toronto, Canada*, 2011.

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

[24] S. Jha and M. Hassan, "Java implementation of policy-based bandwidth management," *International Journal Network Management 2003*.