Using Programmable Network Management Techniques to Establish Experimental Networking Testbeds

Andrew Hughes
PP2 Ross Building
UCL@Adastral Park
Ipswich, IP5 3RE
England
E-mail: a.hughes@cs.ucl.ac.uk

Wolfgang Emmerich
Department of Computing Science
University College London
London, WC1E 6BT
England
E-mail: w.emmerich@cs.ucl.ac.uk

***Abstract*— The way in which research groups evaluate router software (QoS and routing components, for example) seems to be restricted to methodologies using mathematical modelling and simulation techniques. We believe that an experimental methodology is rarely used as the deployment of custom routing software to a testbed comprising multiple routers is a non-trivial task that is beyond the scope of most network research projects. This project intends to make experimental methodologies more accessible to researchers by using programmable networking techniques and by building a management system for a network testbeds.**

I. Introduction

Over the last three decades, a great deal of research has been done relating to computer networks. This has led to the birth of the Internet which enables millions of devices to efficiently communicate. Router devices are used to provide a communications framework to transport data between the computers connected to it. ARPANET [1], the Internet's predecessor, was originally designed to be a fault tolerant network that would enable military computers to communicate despite the occurrence of router or link failures. However the Internet's usage has evolved principally into a communications platform that is exploited by businesses.

The Internet Protocol (IP) [2], [3] is the core concept that allows data to be transferred between nodes. Routers, which operate at layer three of the OSI network model [4], direct data through the Internet towards the intended destination. In addition, routers are often used to control network congestion using Quality-of-Service (QoS) mechanisms.

There are three methodologies that can be used to investigate the behaviour of the mechanisms implemented in routing software: analytical, empirical and experimental. The analytical methodology, which includes mathematical modelling and simulation techniques, and experimental approaches to network research are discussed below. Empirical research methods are not of interest to this project, and are therefore not discussed.

It seems that the engineering of router software is done primarily using an analytical methodology: probabilistic modelling techniques can be used to assess the performance of an algorithm before implementations of the mechanisms are simulated; this means that behaviour of the potential software components can be evaluated quickly.

It is rarely sufficient for research groups to use mathematical modelling to show that their work will perform well when deployed on a live real-world system. Although this approach serves well as an indication of how algorithms will behave in live systems, more realistic results can be achieved using simulations in addition to mathematical techniques.

It is often the case that simulations are applied to software that is shown to behave well by the analytical investigation in order to obtain more realistic results. In the case of router software, the simulation of a network comprised of devices with prototype software components will give a fairly accurate view of the way in which router software will behave in a live real-world systems. Simulation usually requires that the new router algorithms are implemented into software components that can be incorporated into simulation package such as NS-2. It is highly unlikely that network simulators will be capable of modelling every aspect of a live system, therefore a simulation methodology is not sufficient to categorically show that the proposed router software will behave as expected. Rather, an experimental methodology should be used.

For the evaluation of router components using an experimental methodology, like the simulation approach, algorithms must be implemented into software components. Rather than incorporating these components into a simulated system, they are deployed over a testbed. A testbed is a real network built using real devices (i.e. routers) linked using real physical connections (e.g. twisted pair network cables). By deploying the prototype software over the testbed and configuring it appropriately, the behaviour of the prototype software—and therefore the new algorithms—can be evaluated fairly accurately. This approach has been taken in numerous network projects: for example, the work described in [5] uses a testbed to evaluate a Diffserv [6] implementation. Although it is unlikely that a real network's behaviour will be fully captured by a testbed, we believe that the experimental methodology is as close to real-world behaviour as a research group can get.

In order to use an experimental methodology, research groups are usually forced to build a packet routing fabric around the router software components. The main reason why this is an unacceptable strategy is because the construction of a routing engine is a nontrivial endeavour that is unlikely to be within the scope of the research project. The effort of a research group is better spent on the project in-hand rather than building a system that can be used to test it. It is the opinion of many researchers that the engineering of router software should not be limited to mathematical modelling and analysis using simulations. Evaluation using an experimental methodology is likely to yield more realistic results than analytical or modelling methodologies and should therefore be made more accessible to researchers.

Although programmable and active router technologies can be used for experimental network research, due to the lack of management infrastructure required to carry out large scale investigations, these technologies are rarely utilized. The XORP [7] project combines programmable network paradigms with management mechanisms to allow large programmable networks to be managed; however the system allows only individual nodes to be managed (the reason why this is a problem is discussed later). To our knowledge, the work proposed in this paper is the only project that combines a programmable network technology with a testbed management system with sufficient functionality to allow network researchers to easily use an experimental methodology to investigate router software.

Section II describes the work done in the programmable network community and the value its relevant to this project. Section III discusses issues relating to network management. The way in which our proposed system extends the work done in the Promile project and a description of our system's architecture is described fully in Section IV; the way in which the proposed system will be evaluated is then discussed in Section V. Before we draw conclusions in Section VII, related work is outlined in Section VI.

II. Routers

A. Programmable routers

At the most fundamental level, the behaviour of a commercial off-the-shelf (COTS) router is wholly dependent on the algorithms implemented in its software: the way in which data packets are processed is therefore dependent on software provided by the vendor. This software can be configured. However, it is not possible to change the routers embedded mechanisms.

Programmable routers, the devices used to build programmable networks, differ from COTS routers in that the way in which packets are processed is not limited to the mechanisms provided by the router's vendor. Custom router software can be installed and configured allowing a programmable router to behave in a customized manner.

There are two main approaches to creating a highly customisable testbed: application level programmable networks (herein termed 'programmable networks') and active networks. The behaviour of active networks is controlled by capsules. A capsule is a packet that contains both data and fine-grained processing instructions that describe active router behaviour. This light-weight code can instruct the router how that individual capsule should be processed or configure the active router to influence how other capsules are processed. The behaviour of the router therefore is dependent on management instructions interpreted by the capsule processing engine.

There are a number of active network projects, for example the ANTS [8] and Active Packets [9] projects. Active network projects usually define both the active router architecture and the language syntax to which processing instructions must comply. It is often the case that, due to the fact that capsules are rarely interoperable with standard transport protocols, active networks require an overlay network such as ABone [10] to transport capsules.

Unlike in the active router paradigm, overlay networks are not usually needed in programmable networks. These networks generally process ordinary data packets (IP packets for example) that do not carry processing instructions. Network management instructions are not embedded into individual capsules; rather, the router configuration is determined in a system separate from the packet processing engine. This allows programmable networks to be managed using heavier weight mechanisms then active networks. In comparison to active networks, programmable networks are more suited for the deployment of end-to-end QoS mechanisms.

Programmable and active networks differ from networks constructed using Commercial Off The Shelf (COTS) routers in that they are far more customisable. COTS networks are limited to the functionality provided by the router vendor whereas programmable routers' functionality is not limited to a specific set of mechanisms

There are a set of research projects that are sometimes referred to as programmable but do not comply with our definition of programmable networks. These projects either extend the functionality of routers as in projects such as Genesis [11] and Swichlets [12], or attempt to standardise the interface to the switching fabric as in the case of Xbind [13]. We use the term 'programmable router' to refer to a device that can act as an execution environment for software components that are used in the packet processing procedure. By installing custom software components, the capability of a programmable router can be extended beyond the vendor's specification. We believe that, e.g. Click [14] and Promile [15] meet this definition of programmable networks.

B. Uses of programmable networks

There are numerous reasons why the functionality provided by programmable networks does not usually add value to corporate networks: these relate to performance, complexity and functionality of the programmable routers.

Most programmable routers are significantly slower than COTS routers at performing the most common packet processing tasks. There seem to be two reasons for this: the COTS router software is highly optimised industrial strength whereas most programmable router software is non-optimised research-grade; and programmable networks usually run on Unix systems whereas COTS routers make use of specialised hardware. However it is likely that this performance issue would be overcome if the programmable networks added sufficient value to a corporate network.

By the nature of programmable networks, the complexity of the routers visible to the network administrators is far greater compared to networks constructed using COTS routers. A programmable network administrator defines the workings of the router software components used to process packets whereas COTS router administrators specify which vendor defined processing procedures are used. Clearly, the programmable router management process is far more complex compared to the management of COTS routers; administrators are therefore significantly more likely to incorrectly configure programmable routers. A corporation can overcome this problem by using modules provided by a reputable module vendor, however due to the issue described below, it is unlikely that such a vendor will ever exist.

The final issue impeding the acceptance of programmable networks by industry is due to the network behaviour that corporations require. It seems that, for the vast majority of networks, the functionality provided by COTS routers is sufficient. Few companies require packet processing mechanisms that are not provided by the router vendor. For this reason there is little or no advantage of programmable routers over COTS routers.

It is apparent that programmable networks do not provide sufficient value to be used in the construction of corporate networks, however their use can prove beneficial in network research projects. It seems that most network research projects, specifically those on new protocols and congestion control mechanisms: i.e. functionality intended to be incorporated into router software.

It is clear that programmable network technology is rarely useful in corporate networks, it is however invaluable to network researchers. By making use of programmable routers, investigators can rapidly incorporate software derived from new algorithms into network routers that can be deployed to form a testbed. Researchers are therefore not required to build router software in order to examine its performance using an experimental methodology. Clearly, programmable network technology is ideal for the use in the software system described in this paper which is designed to make experimental methodologies more acceptable to researchers.

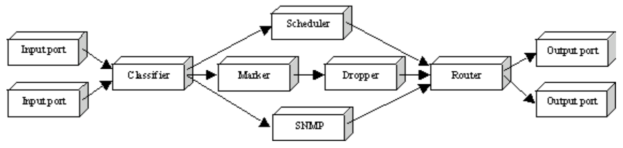

Fig. 1. Example router component graph

C. Router software

Router software can be organized into three categories: the forwarding plane, the control plane and the management plane. The forwarding plane contains queue processing mechanisms that quickly process packets and transmit them on the appropriate output port. The control plane contains functionality that affects the way in which packets are processed at the forwarding plane. Management plane software is used to configure the device: which routing information protocol is used for example.

The functionality of Promile [15] and Click [14] is described by router software components that are associated in order to form a module graph. Figure 1 shows an example module graph. Since the three categories of router software are not apparent in this router configuration model, the programmable routers functionality can easily be controlled. The module graph allows administrators to control the forwarding plane: control and management plane functionality is embedded in the router software modules. The fact that the software components do not have any dependency on other router functionality, enables the programmable router to be configured and extended with few dependency problems.

III. Network management

In order for a programmable network testbed to be accepted by researchers as a valuable tool for the investigation of router software using an experimental methodology, the testbed should be easy to manage. By using a testbed management system that simplifies the task of installing and configuring router software, we believe that we can increase the productivity of a researcher generating result data from the network. In this section, a number of existing management paradigms are outlined and their relevance to our system are discussed.

A. Management of individual routers

Individual COTS routers are usually managed using Command Line Instructions (CLI) or using standards such as Simple Network Management Protocol (SNMP) [16] and the Common Open Policy Service (COPS) [17]. The syntax of CLIs are usually vendor specific: Cisco CLIs are syntactically dissimilar to those of Nortel products, for example. Also, it is often the case that CLI vary from product to product: for example the CLI set of the Cisco 12000 Series Internet Router's is not identical to that of the Cisco 2600 Series Multiservice Platform. SNMP was designed to address the heterogeneity problem of vendor specific CLI and has been adopted by most COTS routers allowing administrators to manage networks without being aware of the architectural differences between routers.

It seems that there are two main approaches taken to manage individual programmable routers. Some projects, Promile [15] and Click [14] for example, define their own configuration languages. These languages are often specific to programmable router architecture, it is therefore rare that a configuration language can be used to manage programmable routers of a different design. The alternate approach, taken by the Alpine [18] and Android [19] projects for example, is to use an existing management system such as DARWIN [20] and CIM [21]. These management systems usually define a set of schemas that can be used to describe software and hardware systems. Administrators configure the programmable routers using a standardised language, allowing heterogeneous issues caused by differing software platforms to be transparently resolved by the management system.

B. Management of multiple routers

In a network consisting of multiple routers, it is unlikely that the router configurations will be homogenous. Routers are configured according to their role, for example the configuration of edge routers may be geared more towards security whereas core routers may be more QoS focused.

From our associations with network providers whose customers include many of the major European banks, we have discovered that corporate networks are usually configured manually using protocol such as SNMP. However it is often the case that networks are monitored by automated systems, and that management instructions are generated by graphical tools operated by network administrators. Using a combination of software, such as Cisco Whatsup and in-house software, network administrators can be notified when erroneous events occur—specifically router failures and software configuration problems. Acting on these notifications, network administrators interact with individual routers to solve router problems.

Since it is the case that routers are usually managed

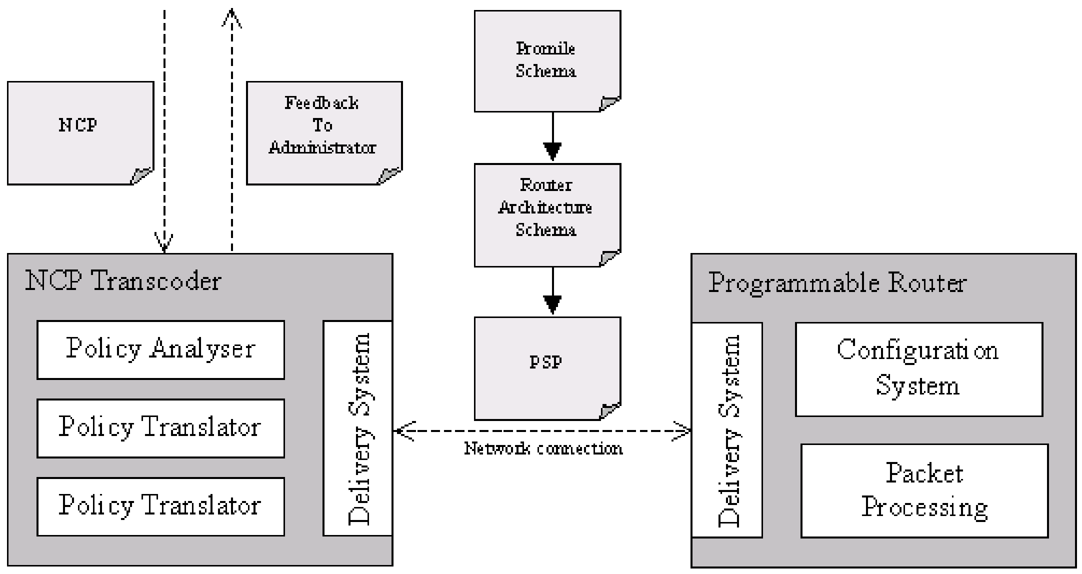

Fig. 2. System architecture

individually, the configuration of a large number of routers is often a laborious error-prone chore in which each router must be individually configured by an administrator. To address this problem, Cisco have released the 2100 Series Intelligent Engine [22], derived form the work done on Directory Enabled Networks [23] by the Distributed Management Task Force. This device can be utilised by administrators to manage either individual or groups of Cisco routers through a graphical interface. Although the way that routers are grouped seems to be fairly limited, the use of this system in a network would significantly reduce the time required to configure a large network, and would reduce the occurrences of problems caused by human errors.

Few corporate networks seem to be configured using automated management systems. This is mainly because there is generally more confidence in the ability of human administrators to configure a network to reflect a company's management decisions compared to an automated network management system. This is especially apparent in telecommunications networks where the way in which telephone calls are routed is specified by human administrators. There are a number of automated network management systems (produced by companies such as Ericsson, BTIgnite and Intelliden, but it seems that these automated client-server management systems generate little interest in the academic research community.

Unlike the server approaches to network management previously discussed, distributed network management poses issues that are of interest to academic researchers. Rather than relying on a centralized management system, projects such as ALAN [24] take a peer-to-peer approach to network monitoring and configuration. Using self-management techniques, an Android network is an autonomous entity that is capable of maximizing its performance and the revenue generates.

Although there are numerous network management systems available, none are appropriate for the management of a testbed comprised of programmable routers. We accept that the above discussed techniques can be used to configure the programmable routers, however these management paradigms are not expressive enough to extend the functionality of programmable routers by installing custom router software modules. It is clear that, in order to provide a software system to allow researchers to evaluate their work using an experimental methodology, a management system capable of configuring programmable routers is required. This project addresses these issues.

IV. System Architecture

As stated in Section I, the aim of this project is to create a software system that enables router software research to be evaluated using an experimental methodology. In this section the software system illustrated in Figure 2 is described. It is our belief that this system will make the experimental evaluation of router software more accessible to researchers. The first subsection discusses the language in which testbed administrators specify programmable network configurations. The subsequent subsection outlines the system that translates

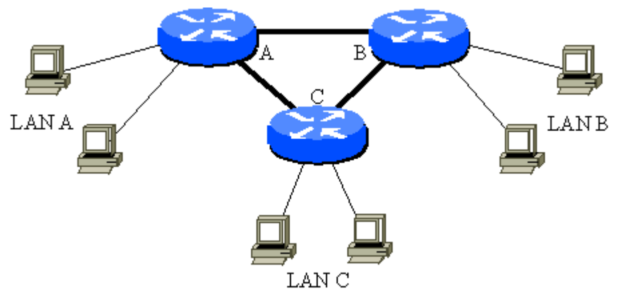

Fig. 3. Example Network

these network scoped configuration policies into configurations that are distributed to individual programmable routers. The final subsection outlines the programmable router architecture.

A. The Network Configuration Policy language

In the proposed system, network administrators control the testbed by specifying the desired network behaviour in a Network Configuration Policy (NCP). The language in which NCPs are written will be defined such that there is a balance between expressiveness and abstraction. The NCP language will be designed specifically for a network administrator to configure the testbed. The language will be abstract enough to disallow the configuration of individual routers to be modified directly, thus forcing network administrators to perform management duties on the network as a complete entity. It is unlikely that a configuration modification of a single router will be sufficient to influence the behaviour of the network, so the language prevents it. The following example illustrated how the configuration of a single router may not result in the network reflecting the desired behaviour.

Consider the network shown in Figure 3. Three Local Area Networks (LANs) A, B and C are connected by three routers which are labelled with respect to the LAN they provide connectivity to. In this example, the initial network configuration allows traffic to travel between all LANs via either two or three routers. It is decided that traffic originating form LAN A must not be permitted to propagate to LAN B: so a network administrator—capable of managing individual routers—configures Router B to drop packets from the Router A. It is likely that—using protocols such as RIP [25], BGP [26] and OSPF [27]—the routers will quickly discover that in order for traffic to travel between LAN A and LAN B, Router C must be used as an intermediary. The behaviour or the network will therefore not reflect the network administrator's desires. By abstracting management instructions so that the behaviour of the network as a whole is described, the network can be made to behaviour as required.

Although it is important for the language in which NCPs are written to have a fairly high level of abstraction, it must be expressive enough to allow custom router software components to be installed onto the testbed. For example, if the testbed used to investigate the behaviour of some Quality-of-Service (QoS) algorithm, the QoS software component must be specified in an NCP. The proposed system will prevent network administrators from modifying the configuration of individual routers, however it is necessary for the NCP language to provide a means of indicating the intended location that some router software components must be installed. It is likely that in our system, rather than using node identifiers, locations will be expressed in terms of a router's role.

An example of the semantic content of an NCP is shown in Figure 4. For the sake of clarity we have formatted the policy in XML. This NCP defines two rules. The first rule will configure the testbed's core routers to process all received IP packets using the 'RED' module available from moduleserver.com via the HTTP protocol. The second rule instructs all of the edge routers in the testbed to process IP packets originating from evil.com with the 'Dropper' module.

B. The NCP Transcoder

In order for an NCP to modify the configuration of the entire testbed, an NCP Transcoder (NT) is used. The NT translates NCPs into Platform Specific Policies (PSPs) that comply with the Promile configuration language [15]. We are confident that the use of this programmable router configuration language will simplify this research project.

The Promile configuration language is an XML [28] based policy language derived from the work done in the Xmile project [29]. Programmable router configura-

```
<NCP>
   <Rule>
      <Target>All core routers</Target>
      <TrafficType>All IP packets</TrafficType>
      <PacketProcessor>RED</PacketProcessor>
      <Repository>http://moduleserver.com/</Repository>
   </Rule>
   <Rule>
      <Target>All edge routers</Target>
      <TrafficType>All incoming IP packets from evil.com</TrafficType>
      <PacketProcessor>Dropper</PacketProcessor>
   </Rule>
</NCP>
```

Fig. 4. Example NCP

tions are described according to an architectural schema, which is in-turn compliant with the Promile schema. Configurations expressed in the Promile language allow network administrators to manage a network in a syntactically homogenous manner, thus transparently bridging issues arising from a network that consists of heterogeneous router architectures. The language does not however conceal semantic architectural difference from the network administrator. The architectural schema is defined by the router vendor: it is used to declared router capabilities. The Promile schema, to which the architectural schema complies, describes that router functionality is implemented as discrete software components (such as packet shapers) that are associated to form a module graph which determines the packet processing procedure.

Given an NCP, the NT will produce one or more PSPs. These PSPs are then deployed to the testbed's programmable routers to control the network's behaviour. The number of PSPs produced by the NT, and their content, will depend as much on the existing behaviour of the network as it does the desired behaviour outlined in the NCP. To illustrate this, consider as an example an NCP that installs some Diffserv component onto a testbed's core routers that affects only packets with the set of properties p. A subset a of the core routers have previously been configured to drop packets with properties p, whereas the remaining routers b do not process packets according to p. Since a must be instructed to process packets with properties p with Diffserv rather than a dropper and b need only be instructed to process packets with properties p with Diffserv, the PSP messages generated by the NT for the routers in set a will be different from those in b.

As we have outlined above, for an NCP to be translated into PSPs, the NT must profile the network to become aware of the current testbed behaviour. The network profile is obtained through analysis of the testbed's router configurations. It will therefore not be necessary to probe the testbed with investigative packets produced by network analysers.

Once the NT has completed the translation of an NCP, the generated PSPs are deployed to routers that constitute the testbed. The NT will deliver the PSPs by utilizing a message oriented middleware (MOM)—such as the Java Message Service (JMS) and Tibco Rendezvous.

During the translation process, problems may occur due to syntactic or semantic errors. If the NT cannot resolve the issues, then the errors will be reported to the network administrator. We assume that if the NT completes the translation and deployment of PSPs, further errors will not occur: i.e. we have full confidence that Promile configuration policies and the architectural schema to which they comply are representative of the routers capabilities and that no unforeseen errors will occur. It is worth noting that this would not be a correct assumption if the management system were being used to manage a corporate network as the occurrence of errors is likely to result in major problems such as loss of revenue.

C. Programmable router

As described in the previous subsection, the system proposed in this paper uses the Promile configuration system to modify the configuration of programmable routers. Although the configuration system can be used to control routing engines from numerous vendors, we intend to use the one developed in the Promile project. To simplify the deployment of new router software components, the testbed will comprise only Promile programmable routers running on a common platform.

Upon receipt of a PSP, the router configuration system processes the policy and makes the appropriate changes to the packet processing engine's software configuration and architecture. Since custom modules can be installed into the router, the architecture of the packet processing engine is dynamic. Software components containing router software are associated to form a module connection graph that controls the way in which programmable routers handle packets.

V. Evaluating the system

To evaluate the research being done in this project, the software system outlined in this paper will be implemented. The complete system will be deployed to form a testbed network which will then be used to show that this system functions as predicted. In order to show that our system can be used to experimentally evaluate forwarding, control and management plane software (as described in Section II-C), components that fall within the scope of these software classifications will be implemented. We will use the RED [30] QoS mechanism to show that forwarding plane software can be investigated using our system. A routing module that implements BGP [26] will be implemented to show that control plane mechanisms can be investigated. Finally, to show the value of the system for experimental research of management plane software, the SNMP [16] management protocol will be implemented into router software components. The deployment of these three investigative scenarios will be done such that our results clearly show that our testbed system is highly reusable due to the powerful configuration management system.

To show that the testbed framework described in this paper is capable of producing more realistic results than simulation techniques, results obtained from simulations and analysis of real world deployment of RED will be compared with results obtained using our system. We are confident that this will show that our system, when used by researchers using an experimental methodology, produces results closer to real world behaviour than analytical approaches to router software investigations.

Existing research that uses testbeds to experimentally investigate router software seems to be very labour intensive: it is certainly the case that the implementation of network mechanisms over a testbed involves the configuration/modification of kernel modules. It will be shown that our system removes the need for an investigator to modify testbed nodes' operating system functionality, resulting in experimental research that is more time efficient and prone to fewer problems produced by human error.

VI. Related Work

It seems to be generally accepted that experimental methodologies are better than analytical approaches to network research. For example, the MBone [31] testbed was designed and implemented in order for researchers to use an experimental methodology to investigate IP multicast techniques. Similarly, the 6Bone [32] testbed was designed for IPv6 research. Since these testbeds were created to perform a specific task, the scope of the research that can be done these testbeds is fairly limited. This limitation was reduced with the creation of the CAIRN [33] testbed which was intended for use by researchers looking at router software. Since the CARIN testbed consists of programmable routers, its functionality can easily be extended. We think that there are two main reasons why this testbed is not suitable for widespread use by network researchers. Firstly, routers are individually configured manually by administrators. Secondly, since most CARIN testbed routers are connected over the Internet, it is not possible for researchers to strictly control the router connections, specifically: network topology, bandwidth, latency and jitter.

The programmable/active network community seems to have begun life when testbeds were not extensible enough to provide a suitable framework capable of supporting diverse network research projects. There are still some research groups focused on active network projects, ANTS [8] and Active Services [34] for example; however, relatively few researchers use active networks to investigate router software. For a survey on various programmable network projects refer to [35] and [36]; these papers contain extensive descriptions and similar works are contrasted.

Programmable routers are generally more flexibility than active networks and are therefore more suited for network research. The key projects in this area are Promile [15], Android [19] and Click [14]. In order to provide a realistic framework for researchers to use to investigate router software using an experimental methodology, the XORP [7] combines the Click router architecture with a management system. The XORP project aims to create a programmable router that will be deployed over corporate networks. These XORP edge routers will allow researchers to investigate the behaviour of router software in a real live network. However, given that the XORP router management system is fairly basic, we believe our system to be superior.

There are a number of projects concerned with the management of programmable routers—most notably NESTOR [37], DARWIN [20], SENCOMM [38], ABLE [39] and ANCORS [40]—however there does not seem to be any management systems designed for networks consisting of multiple programmable routers. PONDER [41], a policy language that can be used to manage a wide variety of computing systems (including networks), is well suited for the management of programmable networks. However, at this stage in our research it is unclear whether or not our system will require such an expressive policy language.

VII. Conclusion

This paper has outlined an area of research that seems to have value in both industry and academia. We intend to build a software system that can be used to manage a testbed comprising multiple programmable routers. By using this software system, researchers investigating forwarding plane, control plane or management plane routing software will be able to efficiently evaluate their work using an experimental methodology. It is believed that this work is novel and that the system outlined in this paper is well suited for the intended task.

Acknowledgment

The authors would like to thank the EPSRC and BTExact for providing funding for this project. Thanks also go to the Promile research group at the Department of Computing Science, University College London, for many stimulating discussions.

REFERENCES

[1] D. Clark, "The design philosophy of the darpa internet protocols," in *ACM SIGCOMM '88*, December 1998.

[2] J. Postel, "Rfc 791: Internet protocol," Sepetember 1981.

[3] S. Deering and R. Hinden, "Rfc 2460: Internet protocol, version 6 (ipv6) specification," December 1988.

[4] I. T. Union, "X.200: Open systems interconnection - basic reference model: The basic model," July 1994.

[5] A. Mohammed, E. Jones, H. Ogier, M. Vouk, and Z. Dwekat, "Diffserv experiments: analysis of the premium service over the alcatel-ncsu internet2 testbed," in *2nd European Conference on Universal Multiservice Networks (ECUMN)*, France, April 2002.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "Rfc 2475: An architecture for differentiated services," April 1998.

[7] M. Handley, O. Hodson, and E. Kohler, "Xorp: An open platform for network research," in *1st Workshop on Hot Topics in Networks (HotNets-I)*, Princton, New Jersey, October 2002.

[8] D. Wetherall, J. Guttag, and D. Tennenhouse, "Ants: A toolkit for building and dynamically deploying network protocols," Ph.D. dissertation, University of Washington, 1998.

[9] A. Kulkarni, G. Minden, R. Hill, Y. Wijata, A. Gopinath, S. Sheth, F. Wahhab, H. Pindi, and A. Nagarajan, "Implementation of a prototype active network," in *First International Conference on Open Architectures and Network Programming (OPENARCH)*, San Fransisco, 1998.

[10] S. Berson, "A gentle introduction to the abone," in *OPENSIG Workshop 2000*, October 2000.

[11] A. Campbell, G. D. Meer, M. Kounavis, K. Miki, J. Vicente, and D. Villela, "The genesis kernel: A virtual network operating system for spawning network architectures," in *Second International Conference on Open Architectures and Network Programming (OPENARCH)*, New York, 1999.

[12] D. Alexander, W. Arbaugh, M. Hicks, P. Kakkar, A. Keromytis, J. Moore, C. Gunder, S. Nettles, and J. Smith, "The switchware active network architecture," *IEEE Network*, May/June 1998.

[13] M. Chan, J. Huard, A. Lazar, and K. Lim, "On realizing a broadband kernel for multimedia networks," in *Worksop on Multimedia Telecommunications and Applications*, Spain, November 1996.

[14] E. Kohler, "The click modular router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, August 2000.

[15] M. Rio, N. Pezzi, H. D. Meer, W. Emmerich, L. Zanolin, and C. Mascolo, "Promile: A management architecture for programmable modular routers," in *OpenSIG 2001*, 2001.

[16] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "Rfc1157: A simple network management protocol (snmp)," May 1990.

[17] D. Durham, K.Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry, "Rfc2748: The cops (common open policy service) protocol," January 2000.

[18] I. Marshall, J. Cowan, J. Crowcroft, M. Fry, A. Ghosh, D. Hutchinson, D. Parrish, I. Phillips, M. Sloman, and D. Waddington, "Alpine - application level programmable internetwork environment," *BT Technology Journal*, vol. 15, no. 2, April 1997.

[19] I. Liabotis, O. Prnjat, and L. Sacks, "Policy-based resource management for application level active networks," in *2nd IEEE Latin American Network Operations and Management Symposium (LANOMS)*, Brazil, August 2001.

[20] J. Magee, N. Dulay, S. Eisenbach, and J. Kramer, "Specifying distributed software architectures," in *5th European Software Engineering Conf. (ESEC 95)*, vol. 989. Sitges, Spain: Springer-Verlag, Berlin, 1995.

[21] "The common information model (cim) standard," distributed Managment Task Force.

[22] "The cisco 2100 intelligent engine," www.cisco.com.

[23] "The directory enabled network (den) standard," distributed Managment Task Force.

[24] M. Fry and A. Ghosh, "Application layer active networking," in *International Workshop on High Performance Protocol Architectures (HIPPPARCH)*, June 1998.

[25] C. Hedrick, "Rfc 1058: Routing information protocol," June 1988.

[26] K. Lougheed and Y. Rekhter, "Rfc 1267: A border gateway protocol 3 (bgp-3)," October 1991.

[27] J. Moy, "Rfc 2328: Ospf version 2," April 1998.

[28] T. Bray, J. Paoli, C. Sperberg-McQueen, and E. Maler, "Extensible markup language (xml) 1.0 (second edition)," October 2000.

[29] C. Mascolo, L. Zanolin, and W. Emmerich, "Xmile: an xml based approach for incremental code mobility and update," *Automated Software Engineering Journal (Special Issue on Mobility)*, vol. 9, no. 2, April 2002.

[30] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, August 1993.

[31] H. Eriksson, "The multicast backbone," *Communications of the ACM*, vol. 37, no. 8, August 1994.

[32] "6bone," www.6bone.net.

[33] ITO, "Collaborative advanced interagency research network (cairn)," in *DARPA ITO Nets PI Meeting*, March 1997.

[34] E. Amir, S. McCanne, and R. Katz, "An active service framework and its application to real-time multimedia transcoding," in *ACM SIGCOMM '98*, Canada, 1998.

[35] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE Communications Magazine*, vol. 35, no. 1, 1997.

[36] A. Campbell, H. D. Meer, M. Kounavis, K. Miki, J. Vicente, and D. Villela, "A survey of programmable networks," *SIGCOMM Computer Communications Review*, vol. 29, no. 2, April 1999.

[37] A. V. Konstantinou, Y. Yemini, and D. Florissi, "Towards self-managing systems," San Francisco, May 2002.

[38] A. J. et al, "Active network monitoring and control: The sencomm architecture and implementation," in *DARPA Active Networks Conference and Exposition (DANCE)*, San Francisco, May 2002.

[39] D. Raz and havitt, "An active network approach for efficient network management," in *International Working Conference on Active Networks*, Germany, 1999.

[40] L. R. et al, "An adaptable network control and reporting system (ancors)," in *DARPA Active Networks Conference and Exposition (DANCE)*, San Francisco, May 2002.

[41] E. Lupu, M. Sloman, N. Dulay, and N. Damianou, "Ponder: Realising enterprise viewpoint concepts," in *International Enterprise Distributed Object Computing Conference (EDOC2000)*, Japan, September 2000.