Netlets: A New Active Network Architecture

Kalaiarul Dharmalingam and Martin Collier
Research Institute in Networks and Communications Engineering
School of Electronic Engineering,
Dublin City University, Republic of Ireland
E-mail: {arul, collierm}@eeng.dcu.ie

Abstract

New conceptual ideas for network programmability have been proposed in the recent past. The Opensig initiative, Active Networks and Mobile Agents are some of the approaches that have been proposed for adding programmability into networks. Active Networks address the problem of mismatch between the rate of innovation in network services and that of the end-user applications. In this model, the switches and routers in the network perform customised computations on the packets flowing through them. With this approach the networks become more readily extensible by allowing new network protocols to be deployed dynamically.

In this paper we advocate a hybrid approach based on Active Networks and Mobile Agents referred to as Netlets for deploying new and modified versions of network protocols. We present the Netlets approach to the provision of network programmability. Netlets are nomadic components that carry service-provisioning code throughout the network. The active nodes in this architecture support runtime environments for processing the Netlet components. We discuss the Netlets network architecture and describe the Netlet Run-time environment. Finally we raise the research issues and the future work that needs to be addressed for realising the Netlets network architecture.

1 Introduction

The explosive growth in networking and computing has generated a need to introduce increasingly complex network services at an accelerated rate. The next generation network is expected to support diverse applications (www, multimedia, telnet), environments (commercial heterogeneous wireline/wireless networks), and workloads (heterogeneous unicast and multicast streams with different quality of service requirements). Consequently, it is clear that the network must play a more active role in supporting the needs of the applications and end users. The data and computer communication networks of today were designed for a fixed service model, thus resulting in a non-flexible network architecture. The rate of change in these networks is restricted by standardisation and compatibility concerns. The result is that introduction of new services occurs much slowly than the emergence of new applications and technologies that benefit from these services. This requires reconciling the perspectives of the computing and telecommunication communities in new dynamically programmable network architectures that support fast service creation through a combination of network aware applications and application aware networks.

Before working on methods to meet the challenge, we need to understand the limitations of existing networks as platforms for new applications. The networking infrastructure currently deployed is a passive layout that carries traffic between end systems with little computation within the network. These traditional networks were designed and customised for a single network service model. This infrastructure is not flexible enough to meet the needs of today's heterogeneous traffic environment.

Network services are readily available functions and utilities that support the connectivity, communications, and control required by applications operating across the network. The level of integration of network services with the network elements (e.g., switches, routers) will determine the network's ability to provide advanced capabilities such as policy management, intelligent object location, and automated fault detection and isolation. By integrating network services into the network, users will benefit from the reduced complexity of the overall network environment. This reduction in complexity can lead to easier management and reduce the overall network cost. The above change in climate has spurred the networking community to consider various approaches to add programmability to the network so as to speed up the evolution of network services.

There has been an increasing demand to add new services to networks or to customise existing network services to match new application needs. Customers are demanding more from the core infrastructure to enable better productivity with flexibility, service differentiation, isolation, privacy, and manageability. This infrastructure is under increased pressure to support complex transport services beyond "simple" packet delivery. The introduction of new services into existing networks is usually a manual, time consuming and costly process. A move from traditional passive networks to dynamically programmable network architectures will support fast service creation and resource management in networks. The ability to program the network would then simplify the deployment of new network services, leading to networks that explicitly support the process of service creation and deployment. This leads to a scenario where control and management points are moved closer to the point of operation. This type of network is referred to as an Open Programmable Network. The most significant of the approaches adapted for realising network programmability are Active Networks [1], Mobile Agents [2] and OpenSig's Initiative [3].

In this paper we advocate a hybrid approach based on Active Networks and Mobile Agents referred as Netlets for deploying new and modified versions of network protocols. In section 3 we present the Netlets approach to the provision of network programmability. Section 4 includes a discussion on the advantages and expected benefits of the Netlets architecture when compared to the other existing Active Network models. In sections 5 and 6 we discuss the Netlets network architecture and describe the Netlet Runtime environment. Finally we raise the research issues and the future work that needs to be addressed for realising the Netlets network.

2 Background

2.1 Active Networks

The concept of Active Networks [1] is relatively new, where a network is not just a passive carrier of bits but a more general computational model. An Active Network may be viewed as a set of active nodes that perform customised operations on the data flowing through them. These networks are active in the sense that nodes can perform computations on, and modify, the packet contents. The Active Network community advocates the dynamic deployment of new services at runtime mainly within the confines of existing IP networks. The Active Network nodes will coexist with the current IP network nodes with no modification to the existing network infrastructure. This simplifies the migration from present passive networks to Active Networks. Such an interoperable network architecture is shown in Fig. 1.

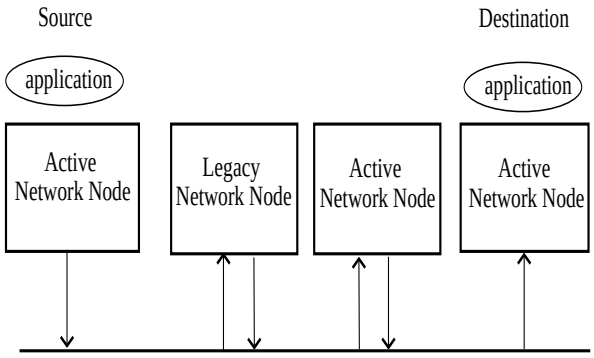

Figure 1: Active Nodes and Legacy Network Nodes

Two different approaches [4], the active packet and the active node approaches have been introduced for realisation of Active Networks. In the active packet approach, the code for processing the packet is carried in the packet itself. The nodes in such a network are also active in the sense that they allow computation at the application layer to take place. The code carried in a packet can be used to process the data of the same packet or can be executed at a node in order to change the state/behaviour of the node. Some examples of the active packet approach are Active IP [5], Smart Packets [6] and M0 [7]. On the other hand, the active node approach allows new protocols to be dynamically deployed at intermediate and end nodes by using mobile code techniques. Some eminent examples of the active node approach are ANTS [8], DAN [9] and CANES [10]. In addition, a hybrid of these two approaches has also been proposed [11].

The Active Packets approach is efficient only when the code to be carried is small. This approach suffers from performance-related problems because of the overhead involved in meeting the safety and security requirements. On the other hand, the active node approach has good performance because security issues are much less than in the previous approach. However, the flexibility of the relevant architectures is limited to the services available at the Active Nodes or to the services supported by the code distribution servers in the network. Neither approach achieves complete dynamic service innovation in the network. A discussion of various approaches can be found in [12].

2.2 Mobile Agents

A mobile agent [2] is an active program that acts on behalf of a user or another program but under its own control. That is, the agent can choose when and where to migrate in the network and can decide on how to continue its execution thereafter. The mobile agents inherit the advantages of all mobile code systems [13], especially the possibility to transport functionality automatically to nodes where it has not been installed before. The difference between conventional mobile code and a mobile agent lies in the inclusion of states i.e. data state and execution state. This state can be transported encapsulated within the mobile agent, whereas conventional mobile code entities are not able to transport state. This feature allows the mobile agent to have the property of autonomous operation. The methodology of mobile agent technology supports encapsulation, program interposition and execution, which make a mobile agent a suitable building block for the construction of Active Networks. In summary the benefits [14] of using the mobile agent paradigm include protocol encapsulation, reducing network traffic, asynchronous and autonomous execution, dynamical adaptation, integrating heterogeneous system, and achieving robustness and fault-tolerance.

3 The Netlets Network

The Netlets network architecture follows the mobile agent paradigm for implementing an Active Network infrastructure. The goal of the Netlets architecture is to build an open, intelligent, customisable, secured network architecture with autonomous, persistent mobile software components- the Netlets. These Netlets are autonomous nomadic components, which encapsulate service-provisioning code that persist and roam in the network independently, providing network services. Netlet components are exchanged on demand by the Netlet nodes to implement customised versions of protocols.

Netlet nodes are can be implemented either as an integrated node or as a distributed node [15]. The integrated Netlet node features a single smart network device (eg. router) with a in-built Netlet runtime environment for service provisioning. The distributed Netlet node consists of a network node with a remote host system, typically a computer supporting the runtime environment for Netlets. In the prototype implementation of the Netlet node we use the distributed node approach.

Another issue that needs to be addressed is the Netlet deployment scheme in the network for service provisioning. There have been many approaches advocated by different research groups including the code-follow-capsule approach in ANTS [8], the code-on-demand approach in DAN [9] and active packets carrying programs in ActiveIP [5]. The use of autonomous nomadic components or Netlets, is advocated for this architecture. In this approach, the code is portable, mobile and autonomous. The expected benefits of this approach are:

Decentralisation and Autonomy: In this architecture the server function for hosting the Netlets is spread across all the nodes of the network. This removes the central point of failures faced with centralised server Active Network approaches [9]. The service code is autonomous and avoids end-user intervention for service deployment within the network [8].

Collective Intelligence: The Netlets approach exhibits collective intelligence of nomadic components for service deployment. In this architecture each Netlet component is an entity by itself and also provides interfaces that allow the Netlet to act as a module in a software architecture for dynamically constructing customised versions of network protocols. This allows building network services in an incremental fashion at Netlet nodes.

Demand-Driven Population of Services: The life of a particular type of Netlet will purely be based on the user demand for that species. On increase of demand for a particular type of Netlets, the required set of species are cloned and dispersed into the network for service deployment. The least used Netlet components would be purged by agent bodies [15] as the demand falls. This leads to a demand-driven population of services in the network.

4 Related Work

The Netlets architecture has common themes with many Active Network approaches exploiting mobile code. The most notable approaches featuring mobile code are ANTS [8] and the DAN [9] approaches, from which many other prototype extensions have been built [17, 18, 19]. The ANTS model takes on the code-follows-capsule approach. This scheme may fail in the case when the source node is attached via a wireless link, because wireless links, unlike most links in common wireline networks, are not necessarily bi-directional. This approach is more tightly coupled with the end user thus making new network services slower to evolve. This is mainly because the end-user is to cache the required code for service provisioning of their packets over the ANTS network. In contrast, the Netlets architecture does not exhibit such shortcomings, because user nodes are not involved in component caching. The DAN approach follows the code-on-demand paradigm and hosts the services in a centralised server, referred as the

code server. These servers become bottlenecks when operating over wide area networks and become points of failures thus resulting in degrading the overall performance. The Netlets architecture disperses this problem by spreading the network services across all nodes in the network.

5 Netlet Node Architecture

The core of a Netlet Node is the Netlet Run-time environment (NRE). The NRE to a Netlet node is like a kernel to an operating system. A Netlet node could be logically divided into two distinct layers, the lower layer represents the physical network node and the top one represents the NRE. A virtual machine representation is used as a shim between these two layers. In our prototype implementation we use the JVM [16] for this purpose. We have chosen Java as the programming language for implementing the Netlets network architecture because it offers a platform-neutral byte-code representation of service code and because of the likely emergence of higher performance compilers and runtime systems for use in real-time systems. Also, the powerful expressiveness of the language, for example object-orientation, libraries for communication, multithreading, and dynamic code linking/loading allows us to easily construct programs from modular components for handling application-specific protocols. The architecture of the Netlet node with the NRE and its components is shown in Fig. 2. It has the following features:

Fine-grain code mobility: This allows us to build incremental network services at the network nodes from modular mobile components.

Flexible, modular and extensible architecture: The proposed design of the NRE is shown in Fig. 2. The NRE layer is a lightweight software layer running on a virtual machine providing only the basic function of network communication and code mobility. We propose to build the other required features (policy, security, accounting etc.,) as add-on modules to the NRE layer. This modular based design makes the system extensible and adaptable. For example if the security-implementing algorithm is found deficient, a new algorithm can be readily installed. This facilitates and enables the system to evolve continuously based on changing requirements.

Real-time operation characteristics: The nomadic mobile code, Netlets, will operate on resource constrained network devices. These devices must be hard real-time systems. Thus it is desirable to have a small code base, and to use as few threads for operation. By decoupling the core mobility-implementing layer from other features like policy, security, accounting etc., the environment will

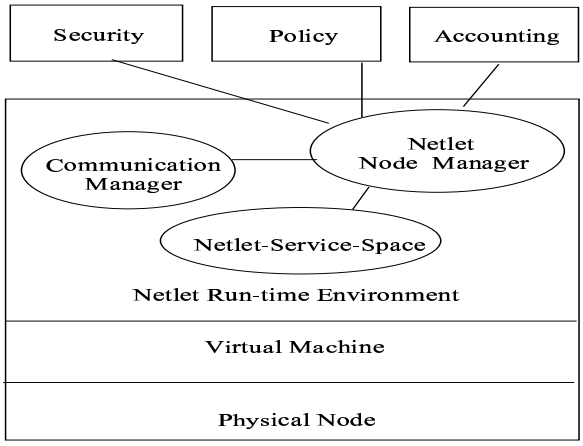

Figure 2: Netlet Node Architecture and Run-time Environment

be customisable to support only required features at a network node. Thus for a Netlet node operating in a network domain for mere packet routing the add-on features like policy and security implementation could be removed or customised according to the need at the particular node.

6 Components in a Netlet Run-time Environment

We follow a modular based design for building the NRE. This allows customisation of the run-time environment as the design evolves. The major components that form the core of the NRE are: a Netlet Node Manager, Netlet-Service-Space and a Netlet Communication Manager.

The Netlet Node Manager is the central module that coordinates the working of the NRE. It records all the Netlets components residing at the Netlet node with their handler specifications. Handlers specify the methods for packing and unpacking Netlets in an NRE. This manager module also provides interfaces to all the add-on modules for implementing their services at the Netlet node. This component also manages the node resources (memory, CPU cycles), access levels (routing table entries, communication to other resident Netlets) allocated to Netlet groups implementing application specific protocols.

Netlet-Service-Spaces allocates physical space and resources for Netlets execution. This environment provided for execution of the Netlets is a sandbox type environment to prevent malicious Netlets from accessing and modifying state of the Netlet node. The Netlet-service-space component accounts for resource usage by the Netlets and reports

them to the Netlet manager. This component also stores Netlet specific handler codes and soft-states of Netlet components during service provisioning. We plan to use this component to provide standard interfaces for communication between Netlets of different sessions. This would prevent malicious Netlet codes attempting to access and change state of properly functioning Netlets.

The NRE Communication Manager provides socket based communication support for the NRE environment for exchanging Netlets on demand among the Netlets nodes in the network. This manager creates communication session for each network service running in the Netlet node.

In addition to supporting the Netlet Run-time Environment (NRE) for portable code, the Netlet node will support the following features for continuous service provisioning:

- Support mobile code communication;
- Identify user requests for protocol features;
- Initiate Netlet discovery ;
- Security validations on Netlets;
- Support dynamic code loading/linking;
- Resource management among Netlet sessions; and
- Provide persistent storage and Account for the resource usage by the Netlets.

7 Conclusion and Future Work

The need for open programmable dynamic network architectures that allow fast protocol evolution in the network has been described. We also discussed various architectures and their shortcomings in realising such networks. We then showed the mobile agent paradigm as a promising candidate for implementing Active Networks. We introduced the Netlet network architecture and stated the major differences and expected benefits over the other existing Active Network models. The modular based architecture of the Netlet node was described with emphasis on the overall design features. The current work is involved in implementing the Netlet node. The second phase of work will involve designing network services for evaluating the prototype implementation. We will also investigate mobile code discovery schemes and the caching strategies that could be adopted in such networks for efficient functioning. Based on these schemes we will then evaluate and test the prototype model for functioning with real-time traffic. This should result in a robust and stable platform for the rapid deployment of new services and applications in tomorrow's Internet.

References

[1] DARPA Active Network Program, http://www.darpa.mil/ito/research/anets/projects.html, 1996.

[2] Nwana H.S, Software agents : An Overview, Knowlegde Engineering Review, vol 11, no. 3, pp.205-244, Oct/Nov 1996.

[3] OPENSIG Working Group, http://comet.columbia.edu/opensig/

[4] D. L. Tennenhouse, J. M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden, A Survey of Active Network Research IEEE Communications, Magazine, Vol. 35, No. 1, pp80-86. January 1997.

[5] David J. Wetherall and David L. Tennenhouse, The ACTIVE IP OPTION,In the 7th ACM SIGOPS European Workshop.

[6] Beverly Schwartz, Wenyi Zhou, and Alden W. Jackson, Smart Packets for Active Networks, BBN Technologies, Jan. 1998.

[7] Albert Banchs et al., M0 Architecture: Multicasting Multimedia Streams with ActiveNetworks, ICSI technical report 97-050.

[8] D. J.Wetherall, J. V. Guttag, and D. L.Tennenhouse, ANTS: Architecture for Building and Dynamically Deploying Network Protocols, Proc. IEEE OPENARCH 98, April 1998

[9] Dan Decasper and Bernhard Plattner, DAN: Distributed Code Cashingfor Active Networks, Proc. IEEE INFOCOM 98, San Francisco, CA, 29 March2 April 1998.

[10] Composable Active Network Elements: Lessons Learned. E. Zegura and K. Calvert, Presented at DARPA AN PI Meeting, Portland, OR, June 2000

[11] Y. Yemini and Sushil da Silva, Netscript:Towards Programmable Networks, Proc. IFIP/IEEE International Workshop on Distributed Systems, Operations, and Management, LAquila, Italy, 1996.

[12] Konstantinos Psounis, Active Networks: Applications, Security, Safety, and Architectures, in IEEE communications survey, First Quarter 1999.

[13] Gian Pietro Picco, Understanding Code Mobility, Politecnico di Torino, Italy, Tutorial at ECOOP98, 22 July 1998

[14] Danny B.Lange, Mitsuru Oshima, Seven Good Reasons for Mobile Agents, Communications of the ACM, Vol. 42, No. 3,1999.

[15] M. Collier, Netlets: the future of networking?, First IEEE Conference on Open Architectures and Network Programming, San Francisco, CA, USA, April 3-4, 1998.

[16] The Java Virtual Machine Specification, http://java.sun.com/docs/books/vmspec

[17] Erik L. Nygren, Stephen J. Garland, and M. Frans Kaashoek, PAN: A High-Performance Active Network Node Supporting Multiple Mobile Code Systems IN PROCEEDINGS IEEE OPENARCH99, MARCH 1999

[18] Gsli Hjlmtsson , The Pronto Platform - A Flexible Toolkit for Programming Networks using a Commodity Operating System, OPENARCH 2000.

[19] D. Decasper, Z. Dittia, G. Parulkar, and B. Plattner, *Router plugins: A software architecture for next-generation routers*, IEEE/ACM Transactions on Networking, Vol. 8, No. 1, Feb. 2000, pp. 2-15.