Active Networking Architecture

Akshat Gupta
Department of Computer Science
Shiv Nadar University
akshat1706@gmail.com

Shivam Sharma
Department of Computer Science
Shiv Nadar University
shivamsharma4723@gmail.com

Akarsh Kala
Department of Computer Science
Shiv Nadar University
akarshkala@gmail.com

***Abstract*—**Active networks enable their end users to insert tailor made or customized programs into the nodes included in the network. A concept that we are trying to explore is the capsule concept - where we try to replace traditional packets in a passive networks with "capsules" - segments of programs that get executed or implemented at each network node/router that they switch or travel/traverse.
Active architectures allow a massive increase in the the complexity of the computation that is allowed internally in the network. These computations can enable modern and different applications, specially those based on information fusion, and other services that leverage networking based storage and computational power. Further, this network architecture will accelerate the rate of innovation by deconstructing rigid networking from the hardware beneath and allowing new programs and services to be loaded into the infrastructure as and when required.

Keywords—active networks, computer architecture, interoperability, encapsulation.

I. WHAT ARE ACTIVE NETWORKS

Active Networks is a relatively new concept, where a network is not just a passive carrier of bits but a more general computation model. It is a highly programmable networks that perform computations on the user data that is passing through them. Active Network may be simplistically viewed as a set of 'Active Nodes' that perform customized operations on the data flowing through them. Traditional data networks provide a transport mechanism to transfer bits from one end system to another, with a minimal amount of computation (e.g., header processing and signaling). In contrast to that active networks not only allows the network nodes to perform computations on the data but also allow their users to inject customized programs into the nodes of the network, that may modify, store or redirect the user data flowing through the network. These programmable networks open many new doors for possible applications that were unimaginable with traditional data networks. For example, there may be a video multicast session where at every node the video compression scheme is modified, based on the computation done by that node and depending on the network bandwidth available.

II. LEAD USERS

Lately, there has been a significant interest in agent technologies, which allows code which is mobile to traverse from server to clients and vice-versa. Implementing active networks as an architecture will bridge this gap by performing/dispatching computation as and when required. There are a variety of users that need/implement this computational functionality in very specific forms. within the network. These include the developers of :

- Firewalls, are regularly located at administrative boundaries.
- Web proxies and various other similar services such as the DNS and some multicast routers.
- Nomadic gateways, which are placed in proximity of edges of the networks where there are considerable disparities and discontinuities in the bandwidth that is available.

A. Firewalls

They make use of filters that determine which packets pass and which should be blocked according to the administrators wishes. They implement application and user specific functions, after the basic packet routing.This process could be implemented in Active networks by automation by allowing application and network packets from the approved sources or vendors to authenticate themselves to the firewall and inject appropriate modules in it.

B. Web Proxies

This is an classic example of application-specific service that is specifically made for serving and caching of the World Wide Web Pages. We can confidently say that we can implement active network and its related technology here for web caching because a large and considerable amount of web pages are computed dynamically and are not susceptible to the traditional (passive) styled caching. Having this in mind we can suggest the development of web proxy schemes that support "active" caches that store and execute the programs that are capable of generating web pages.

C. Nomadic Computing

Mobility is a big factor for developers and researchers. A nomadic router is a router that is interposed between an end system and the network. This is an classic example of application-specific service that is specifically made for serving and caching of the World Wide Web Pages. We can confidently say that we can implement active network and its related technology here for web caching because a large and considerable amount of web pages are computed dynamically and are not susceptible to the traditional (passive) styled caching. Having this in mind we can suggest the development of web proxy schemes that support "active" caches that store and execute the programs that are capable of generating web pages. and/or invoke additional security, such as encryption, when operating away from the home office.

Similarly, "nomadic agents and gateways" are nodes that support mobility. They are located at strategic points that bridge networks with vastly different bandwidth and reliability characteristics, such as the junctions between wired and wireless networks.

New Application Domains

There is an untapped reservoir of applications that require sophisticated network-based services to support the distribution and fusion of information.

Information fusion is an example of a domain that may leverage interposed computation. Applications such as sensor fusion, simulation and remote manipulation, allow users to "see" composite images constructed by fusing information obtained from a number of sensors. Fusing data within the network reduces the bandwidth requirements at the users, who are located at the periphery of the network. Placing application-specific computation near where it is needed also addresses latency limitations by shortening the critical feedback loops of interactive applications.

III. APPROACHES

Active networks are highly programmable networks that perform computations on the user data that is passing through them. We distinguish two approaches to active networks, discrete and integrated, depending on whether programs and data are carried discretely, i.e., within separate messages, or in an integrated fashion.

A. Programmable Switches - Discrete Approach

When a packet arrives, its header is examined and a program is dispatched to operate on its contents. The program actively processes the packet, possibly changing its contents. A degree of customized computation is possible because the header of the message identifies which program should be run – so it is possible to arrange for different programs to be executed for different users or applications. The processing of messages may be architecturally separated from the business of injecting programs into the node, with a separate mechanism for each function. Users would send their packets through such a "programmable" node much the way they do today.

B. Capsules - An Integrated Approach

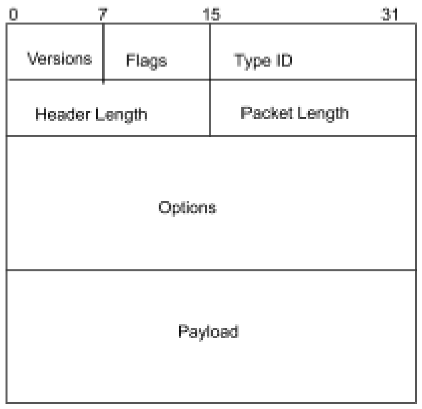
A more extreme view of active networks is one in which every message is a program. Every message, or capsule, that passes between nodes contains a program fragment (of at least one instruction) that may include embedded data. When a capsule arrives at an active node, its contents are evaluated, in much the same way that a PostScript printer interprets the contents of each file that is sent to it.

Bits arriving on incoming links are processed by a mechanism that identifies capsule boundaries, possibly using the framing mechanisms provided by traditional link layer protocols. The capsule's contents are dispatched to a transient execution environment where they can safely be evaluated. We are assuming that programs are composed of "primitive" instructions, that perform basic computations on the capsule contents, and can also invoke external "methods", which may provide access to resources external to the transient environment. The execution of a capsule results in the scheduling of zero or more capsules for transmission on the outgoing links and may change the non-transient state of the node. The transient environment is destroyed when capsule evaluation terminates.

IV. PROPOSED PACKET FORMATS

A. ANEP (Active Network Encapsulation Protocol)

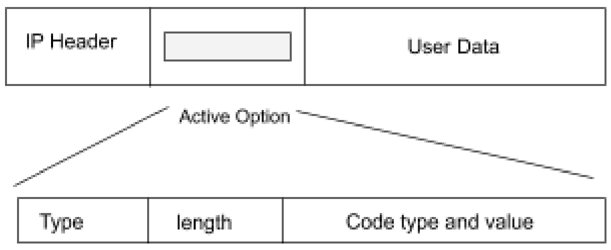
Researchers engaged in the field of active networking outlines a plausible mechanism for encapsulation of active network frames. This protocol would allow the use of existing network architecture in the deployment of active networks. This mechanism is a generic, extensible and ensures co-existence of different execution environments. The proposal calls for an active network frame format that would have the programs to be executed integrated into its payload. These programs would be executed at the receiving node in an environment specified by the protocol. ANEP would allow specifications of various options in its header that includes but is not limited to authentication, confidentiality and integrity.



Here, the 8-bit version field indicates the header format in use (current version being 1). Only the most significant bit of the 8-bit flags field is used in version 1 and it indicates whether the node should forward the packet or discard it, if it does not recognize the following Type ID field. The Type ID field would indicate the evaluation environment of the message. A proposed ANANA - Active Networks Assigned Numbers Authority would assign the Type IDs, with 0 being reserved for error messages and possible future network layer information. This is followed by header length and packet length fields, both in units of 32-bit words. Each option in the options field has a 2 bit flag, followed by a 14-bit Option Type field (again controlled by ANANA, however, a 0 in the flag field would mean a non-standard option). A 16-bit option length followed by the Option Payload follows this. The robustness of the proposed protocol ensures that implementing a new protocol or evaluation environment would mean only assigning a new Type ID and a new Option Type.

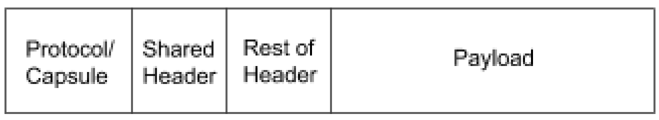
B. ACTIVE IP

Wetherall and Tennenhouse proposed an extension to the IP protocol that would retrofit active capabilities to the existing Internet.



As the seen in the above figure, the Active IP option field provides a mechanism for embedding a program fragment in an IP datagram. These fragments in the options field are then evaluated and executed at every router along the path. Since, the ACTIVE option is in the payload of the packet; the legacy router would not even see it and forward the packets to active routers. At these active routers, the Active Option code would be invoked for the routers to execute.

C. ANTS



ANTS use mobile code techniques that enable new protocols to be deployed at both end systems and intermediate nodes. ANTS use the encapsulation approach with a code distribution mechanism. The protocol incorporates code-forwarding routines into each node, which acquires the application specific routines by a code distribution scheme. The forwarding routines are transferred to related capsule types, i.e. a code group by the code distribution system. The capsule format is shown in the following picture.

Each capsule would carry an identifier for its protocol type, which is described from the code description of the protocol. The identifier is based on the fingerprint of the protocol code, which reduces significantly the risk of protocol spoofing. The remainder of the capsule comprises of a shared header common to all capsules in the code group and then the unique header. The protocols are executed within a restricted environment with limited access to the shared resources, thus safeguarding it from runaway protocols. An initial set of primitives is loaded to the nodes. The node provides with an execution model that support issues like network security and resource management. The protocol then has a code distribution mechanism that propagates the program definitions to where they are needed. The scheme works as follows:

- Firstly, the capsule identifies their type and protocol
- Then, when a capsule arrives at the node, it checks its cache to see if the required code is present, if not it request its previous node to send the code to the current node.
- When the previous node receives a load request that it may answer, it sends immediately
- When the current node receives the load response, it loads it into its cache and performs the required action on the waiting capsule.

This is an efficient and robust scheme that would allow the nodes to take advantage of active networking.

V. THE NETLET NETWORKS

A. THE MOBILE AGENT

A mobile agent 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 , especially the possibility to transport functionality automatically to nodes where it has not been installed before. 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 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.

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. 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.

In this approach, the code is portable, mobile and autonomous. The expected benefits of this approach are:

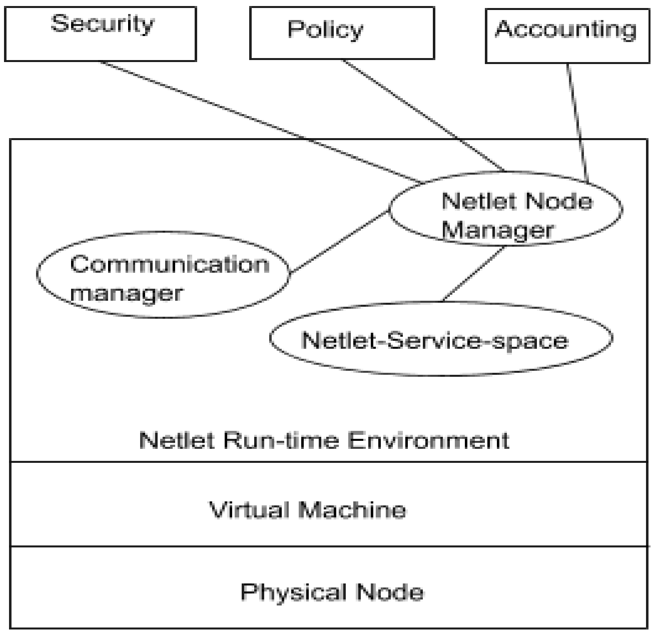
- **Decentralisation and Autonomy**: In this architecture the server function for hosting the Netlets is spread across all the nodes of the network. This removes the central point of failures faced with centralised server Active Network approaches.
- **Collective Intelligence**: In this architecture 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.

B. NETLET NODE ARCHITECTURE

The core of a Netlet Node is the Netlet Runtime 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 for this purpose. We have chosen Java as the programming language for implementing the Netlets network architecture because it offers a platform-neutral bytecode 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 figure after this. 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 the figure. 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 addon 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. 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 be customisable to support only required feature at a network node.

VI. COMPONENTS IN A NETLET RUNTIME 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, NetletService-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.

VII. WHY HASN'T THIS BEEN IMPLEMENTED YET

One of the challenges of active networking has been the inability of information theory to mathematically model the active network paradigm and enable active network engineering. This is due to the active nature of the network in which communication packets contain code that dynamically change the operation of the network. Fundamental advances in information theory are required in order to understand such networks.
The approach we are proposing synthesizes a number of technologies: programmable node platforms, component-based software engineering, and code mobility. A few "programmable" networks have been developed in the past, and suggestions for object-based approaches to network implementation surface every few years. However, the previous work has not leveraged code mobility within the network, let alone within the context of each and every capsule or packet.

SUMMARY

Active Networks seem to break the architectural rules which are holded inviolate by the generally believed wisdom. The past successes with packet approaches, like the Internet, are believed to be the basis on which they are built. The same time they relax some architectural limitations, which may now be the artifacts of the older generations of hardware and software technology.
All this time we have been served pretty well with the passive network architecture that emphasise packet-based end-to-end communication. However, computation within the network is already happening. It would be out of our fortune if network architects were the last to notice.
It's time to explore newer models of architecture, such as these active networks, that incorporate interposed computation. These efforts would enable new generations of highly flexible networks, ones which would fasten the pace of infrastructure innovation.

CONCLUSION

In this paper we have described our vision of an active network architecture that can be programmed by its users. We expect that active networks will enable a range of new applications in addition to the lead applications that already rely on the interposition of customized computation within the network. However, we believe that this work will also have broader implications, on how we think about networks and their protocols; and on the infrastructure innovation process.

REFERENCES

[1] D.J.Wetherall. Active Network vision and reality, *Published as Operating Systems Review, Dec.1999.*

[2] M. Brunner. Active Networks and its Management, *Germany.*

[3] Sohail M. Active Networks - A Survey, *July 2000.*

[4] David L.Tennenhouse and D.J.Wetherall. Towards an Active Network Architecture.

[5] A.Galis, B.Plattner, JMSmith. A Flexible IP Active Networks Architecture, *IWAN 2000 Conference.*

[6] B.Bhattacharjee, K. Calvert, E. Zegura. Implementation of an Active Networking Architecture.

[7] U Legedza, D.J.Wetherall, J.Guttag. Improving the Performance of Distributed.
Applications Using Active Networks, *In IEEE INFOCOM, San Francisco, April 1998.*

[8] Kalaiarul D, M.Collier. Netlets: A New Active Network Architecture.

[9] Karthikeyan S, Vincent AS, Stephen W.Keckler, DBurger. Routed Inter-ALU Networks for ILP Scalability and Performance, *21st International Conference on Computer Design.*

[10] B. Bhattacharjee, K. Calvert, and E. Zegura. An architecture for active networking. *In Sub-
mitted to IEEE Infocom '97, 1997.*

[11] S. Bhattacharjee, K. L. Calvert, and E. W. Zegura. On active networking and congestion.
Technical Report GIT-CC-96-02, College of Computing, Georgia Institute of Technology, 1996.

[12] D. L.Tennenhouse and D. J. Wetherall. Towards an active network architecture. *In Multimedia Computing and Networking '96, January 1996.*