

Mobile Agents for Active Network Management

By Rumeel Kazi and Patricia Morreale
Stevens Institute of Technology
Contact: rkazi,pat@ati.stevens-tech.edu

Abstract-Traditionally, network management systems have been based on client/server technologies comprised of distributed static agents (servers) and a centralized manager (client). The behavior of such systems has proven to be difficult to modify and/or extend. The use of mobile software agents has been introduced as an alternative to traditional centralized network management techniques. Mobile software agents provide a decentralized approach to Network Management issues because of the distributive nature of the agents themselves. Mobile software agents can be defined as autonomous software components operating within a network. They perform specialized tasks by traversing the network and working independently or in conjunction with other software components. The problem of bottlenecking or network congestion associated with a centralized manager has been shown to be non-existent with the use of mobile software agents. Mobile Agents in a broad sense are programs that represent a user in a computer network, and are capable of migrating autonomously from node to node, to perform some computations on behalf of the user. In this paper, they have been introduced in an active network environment to dynamically make required changes in the network and also make dynamic network service innovation attainable.

I. INTRODUCTION

The demand for network change, including support for advanced services and features, has led to the development of active network [1]. An active network permits applications to inject customized programs into network nodes. This permits faster protocol innovation by making it easier to deploy new network protocols [2, 3], even over the wide area [4]. In this paper, we argue that the ability to introduce active protocols offers important opportunities for end-to-end performance improvements of distributed applications. Furthermore, a network can be configured to work with many protocols that are application dependent unlike the traditional networks running a single protocol. The ability to have a network adapt to the application needs, in real-time, rather than the traditional network approach of support for a single protocol, greatly increases the potential attributes and services of network applications [9].

To support this active network environment, three active task groups called Active Network Service/Application Task (AST), Active Network Control Task ACT, and Active Network Management Task (AMT) which reside in routers/switches with specific service, control, and management responsibilities. These executables let network switches and routers perform application-specific functions and apply appropriate algorithms and parameters for control and management of services. In this paper, the Active Network principles based on the active tasks and active packets are presented and an active network management scenario is discussed and analyzed.

An Active Packet can be classified as

- Service Packet that may contain data in a typical file transfer application.
- Control Packet that may contain control information to adapt to increasing network traffic on a link or
- Management Packet that contains node management information for e.g. buffer space, bandwidth or signal-to-noise ratio.

The objective is to design a protocol that works well in such an Active Environment where all the three types of packets are involved. A robust protocol that can be used to increase the performance of an Active Network is presented here. In this paper we explain the protocol, the capsule format and the use of mobile agents [7] to distribute the status of the nodes in the active environment.

II. NETWORK TASK GROUPS

Three active network task groups called Active Network Management Task, Active Network Control Task, and Active Network Service/Application Task, which reside in routers/switches with specific service, control, and management responsibilities are used. We have shown that these task groups comprise the basis for the implementation of delegated network management [8] for active networks and provide a basis for the future expansion of active network functionality.

Each group entails a specific requirement and communicates through Active Packets (Capsules). These AP's, namely, Active Service / Application Packet (ASP), Active Control Packet (ACP), and Active Management Packet (AMP), carry executables for service control and network management purposes.

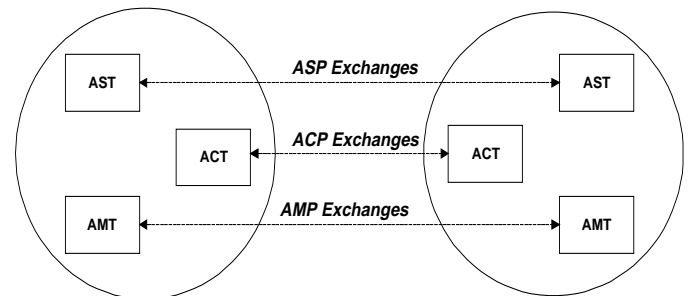


Fig. 1. Exchanges among active tasks are through active network packets: ASP, ACP and AMP

III. GENERAL DESIGN

The active network toolkit, ANTS [2] design is guided with three simple goals in mind, each of which represent a more

flexible form of innovation than is currently achieved in the Internet.

1. The nodes of the network should support a variety of different network protocols being used simultaneously.
2. The nodes of the network should support a variety of different network protocols being used simultaneously.

- All the packets in transition in an ANTS network contain a reference to the routine that will be invoked to forward these packets at network nodes. This allows control of the forwarding of the packets but does not give that node any domain over the packets that it is receiving.

TABLE I
ACTIVE NETWORK TASK GROUPS

| Task Group | Action | Response |
|---------------------------------------|--|--|
| Active Service/Application Task (AST) | <ul style="list-style-type: none"> • Obtain channel information from ACT to determine source coding /compression technique. • Determine the number of local connections that can be supported for a given application. • Call connection establishment processing based on call type and QoS requirements. • Dynamically determine whether a service/application should be supported locally or referred to another node. • Error detection/ correction. | <p>Execute the appropriate program to realize them.</p> <p>Establish/Deny new connections.</p> <p>Either execute a program or make a call to remote system.</p> |
| Active Network Control Task (ACT) | <ul style="list-style-type: none"> • Respond to AST commands • Resource allocation as demanded by the AST. • Determine routing for each application currently active and for new connection requests. • Determine node's overload status. • Routing table update based on data received from other ACTs and local AMT. | <p>Change channel rate in response to AST's QoS requirements and subsequent AST commands.</p> <p>Send appropriate commands to Service / Application upon overload or routing table update.</p> |
| Active Network Management Task (AMT) | <ul style="list-style-type: none"> • Operations, administration, management, provisioning, and billing (OAMPB). • Respond to network alarms, and threshold hits at local nodes and in exchange with other nodes' AMTs. • Collect and exchange with ACT the data to determine whether a connection can be established (e.g. bandwidth, buffer space, delays, and loss) and to determine congestion state and routing. • Collect router /switch processor load conditions (required by ACT to determine overload status and connection control). • Security and Resource management at routers /switch. | <p>Collects, processes, and formulates the network management procedures relevant to its operation. AMT forwards its relevant management data to other active nodes through AMPs.</p> <p>Respond to network alarms, and threshold hits at local nodes and in exchange with other nodes' AMTs.</p> <p>Perform operations, administration, provisioning functions locally, and in concert with other nodes through AMPs.</p> |

3. New protocols should be supported by mutual agreement among interested parties, rather than requiring centralized registration or agreement between all parties.
4. New protocols should be deployed dynamically and without the need to take portions of the network "off-line".

The ANTS model has the following characteristics that can be described as generalized packet forwarding:

- All the packets in transition in an ANTS network contain a reference to the routine that will be invoked to forward these packets at network nodes. This allows control of the

forwarding of the packets but does not give that node any domain over the packets that it is receiving. These forwarding routines are limited in what they can achieve and can thus be written or provided by trusted users. Like most forwarding routines, they are expected to run to completion locally and within a short span of time.

- Individual forwarding routines will do different things at different nodes depending in the local environment. In particular, only a subset of nodes may be appropriate for particular routines, with other nodes performing default forwarding instead.

This approach is novel in that it is intended to be flexible enough to support new protocols, while guaranteeing some level of protection, allocation and performance of shared resources in an untrusted environment. The ANTS model is also connectionless, in that every packet is an independent entity and may refer to its specialized forwarding routine at any place and any time.

IV. THE PROTOCOL

According to our design scenario, we have three types of packets (capsules) moving around in a network of Active Nodes. An active packet sorter like the one shown in Figure 2 resides in each node in the network.

The active packet sorter performs the following tasks:

- Determines the type (ASP, ACP or AMT) of incoming active packet.
- Depending on the type of packet it passes the packet to specific packet handler.

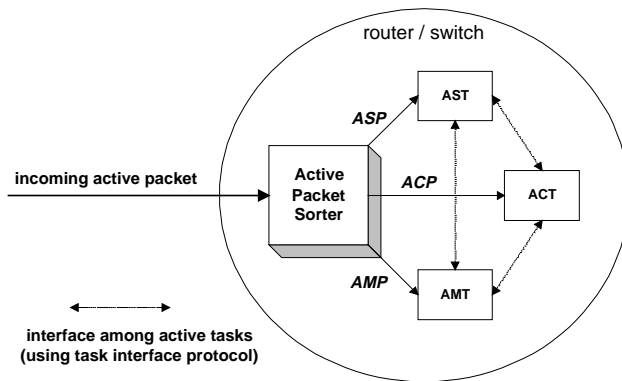


Fig. 2. An Active Packet Sorter in a Router / Switch

The sorter may just pass this packet to a function in the same program or may spawn a process to handle the request. The later approach is better since it does not slow down the sorter from reading the next incoming packets.

V. CAPSULE FORMAT

We implemented the above mentioned protocol using the capsule format shown in Figure 3.

| | | | | | |
|------|--------|------|----|----|----|
| Data | Length | S(n) | SA | DA | SB |
|------|--------|------|----|----|----|

Fig. 3. The capsule format for our protocol

Various fields in the protocol definition above are explained below:

- SB (Sorter Bit): Sorts whether the packet to be delivered is service, control or management packet. This is a two-bit field that determines the packet type. The value of these bits can be :

- 00 – indicating that the active packet is a service packet
- 01 – indicating that the active packet is a control packet
- 10 – indicating that the active packet is a management packet

11 – This combination can be application dependent. For example, it can be a reply from a node to the management node and Data field may contain the node state information.

- DA (Destination Address): Defines the address of the destination station. This is a one-byte field.
- SA (Source Address): Defines the address of the source station, which again is a one-byte field.
- S(n): Specifies the sequence number of active packets and is a six bit field. This field is used when the active packet is a service packet and Data field contain the application specific data.
- Length: Specifies the length of the data in the 'Data' Field.
- Data: Carries the actual data to be delivered at the destination. The value of this field depends on the packet type.
 1. For a service packet it contains the application specific data.
 2. For a control packet it contain control information. This information can be a command from the management node or from a slow receiver or router to a fast sender or router to slow down.
 3. For a management packet it contains the management information at a node like the used bandwidth, available and used buffers, QoS and S/N ratio. Depending on this information a node may update its routing table to route the next outgoing packet through a less congested route. For a centralized management node system, this information may trigger a control information from the management node to be sent to another node(s).

VI. IMPLEMENTATION

We implemented the protocol using ANTS. We started with a simple file transfer application to transfer a file from one active node to another through intermediate nodes working as routers. We implemented slow start algorithm with a window size of 100 capsules. Every capsule except the last one carried 256 bytes of data, one byte of length field, six bits of sequence numbers, one byte each for source and destination addresses.

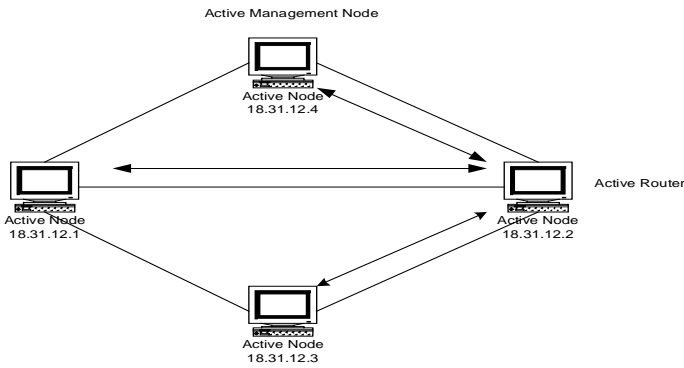


Fig. 4. Active Network

Four nodes were configured to work in the Active Network environment as Active Nodes as shown in Figure 4. The nodes were configured to work as follows:

TABLE II
NODE CONFIGURATIONS

| | |
|-----------------|--|
| Node 18.31.12.1 | Sends Active Service Packets (capsules) to node 18.31.12.3. |
| Node 18.31.12.2 | Receives capsules and routes them on an appropriate link after reading the routes from the routing table. |
| Node 18.31.12.3 | Receives Active Service Packets (capsules) from node 18.31.12.1 and Active Management Packets (capsules) from node 18.31.12.4. |
| Node 18.31.12.4 | Configured as an Active Management Node. It monitors the nodes in the network and sends Active Control Packets when necessary. |

Node 18.31.12.1 sends the capsule shown in figure 5 to Node 3. When the capsule reaches the Node 2, it copies the capsule in its cache and routes it to Node 3.

| | | | | | |
|--------------------------|-----|------|------------|------------|----|
| Data 256 bytes | 256 | S(n) | 18.31.12.1 | 18.31.12.3 | 00 |
|--------------------------|-----|------|------------|------------|----|

Fig. 5. Active Service Capsule in a File Transfer Application

VII. ACTIVE NODE MANAGEMENT AGENT

As shown in Figure 5, Node 18.31.12.4 is configured as an Active Management Node. Periodically it spawns an Active Agent into the network. This agent has destination field value of 18.31.12.2. The length and data fields are both zeros. The sequence number field here is used to indicate the number of hops that the agent has to go through before returning with the results. The management node initializes it with the maximum number of hops for example the diameter of the network. At each hop the value decrements by one.

The Active Management Capsule has the following format:

| | | | | | |
|---------|---|------|------------|------------|----|
| 0 bytes | 0 | S(n) | 18.31.12.4 | 18.31.12.2 | 10 |
|---------|---|------|------------|------------|----|

Fig. 6. Active Management Capsule

When this packet reaches the node 18.31.12.2, the Active Packet Sorter at that node recognizes it to be an Active Management Packet and spawns a process to handle the packet. The process in turn writes the values of available buffers, bandwidth, BER, PER and also updates the Destination Address field value and sends the packet to next node in the list.

The packet forwarded by Node 18.31.12.2 has the following format.

| | | | | | |
|--|------|--------|------------|------------|----|
| Node No - 18.31.12.2 Bandwidth Available Buffers Bit Error Rate Packet Error Rate S/N Ratio QoS | size | S(n)-1 | 18.31.12.4 | 18.31.12.3 | 10 |
|--|------|--------|------------|------------|----|

Fig. 7. Packet forwarded by Node 18.31.12.2

The next node repeats the algorithm and routes the packet. Finally the packet reaches the Management Node where these values are retrieved. The management node may take appropriate steps by sending control packets to nodes to update their routing tables based on current traffic on each links.

VIII. EXPERIMENTAL RESULTS

We implemented the above mentioned protocol on four Sun System running UNIX

A plain text file was transferred from Node 1 to Node 3 with Node 2 as an intermediate router. We simulated a typical network environment. Node 2 was configured in such a way that it did not send one packet out of every ten that were destined for Node 3. It would just store the packet in its cache. The cache was configured to buffer 1024 incoming and outgoing capsules. Node 4 was again configured to act as a centralized Active Management Node. The node would send out a management agent upon initialization.

We implemented the Client Side TCP [6] that involved network level caching [5]. Analysis shows a reduction of up to 88 % in redundant bandwidth and redundant packet latency with network node caching [6].

As shown in figure 8, Node 3 requests a capsule from Node 1 but the capsule gets lost on the way. Node 3 times out and sends the request again. Since, the lost packet would have been cached in an intermediate node on the way, the node itself would send the cached capsule. In this model, it is assumed that request capsules are routed along the same network path as the data capsules.

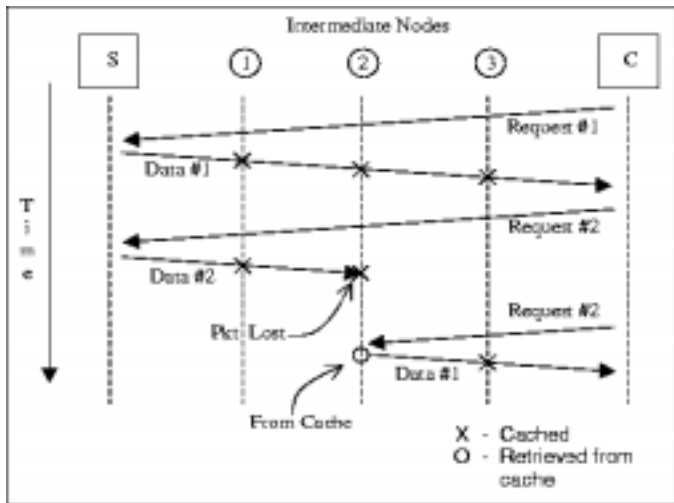


Fig. 8. Client Side TCP

The agent comes back to Node 4 with the following information about Node 2 and Node 3.

TABLE III
AGENT RESULTS

| Parameter | Node 2 18.31.12.2 | Node 3 18.31.12.3 |
|-------------------|----------------------|----------------------|
| Bandwidth | 172 caps / sec | 165 caps / sec |
| Available buffers | 94 / 1024 (used) | 84 / 1024 (used) |
| Bit Error Rate | 0 / capsule | 0 / capsule |
| Packet Error Rate | 1 / 10 capsules | 1 / 10 capsules |

The management node evaluates the results and it now knows that there is some problem on the link connecting node 2 and node 3. It sends appropriate control packets to determine the cause of this loss. If the problem persists, another agent is dispatched by the Management Node that goes to each node whose routing table is to be updated. The agent takes the following steps to update the routing tables:

1. Copy the contents of the routing table which in our case is a plain text file.
2. Make required modifications to this file.
3. Overwrite the old file with the modified one.
4. While overwriting the new file, the agent locks the file read operations for this file.

IX. FUTURE RESEARCH

Although our experimental network was composed of only four nodes, some conclusions can be made about the scalability of the proposed protocol and the decentralized network management scheme [10]. Future work includes a better understanding of how to manage routing table updates, and the potential for associated latency in this decentralized environment. Furthermore, the experimental results here involve a network where management nodes proactively gather information about their operating environment. In the future, a report-driven environment will be considered and support for highly mobile networks will be addressed.

REFERENCES

- [1] [Tennenhouse 97] David L. Tennenhouse, Jonathan 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
- [2] Wetherall, Guttag, Tennenhouse, "ANTS: A Tool kit for Building and Dynamically Deploying Network Protocols", Submitted to IEEE OPENARCH'98, San Francisco, April 1998. Available: <http://www.tns.lcs.mit.edu/publications/openarch98.html>
- [3] David J. Weatherall, "Developing Network Protocols with the ANTS Toolkit" Design Review, August 1997
- [4] Introducing New Internet Services: Why and How, IEEE Network Magazine, July/August 1998
- [5] Using Network Level Support to Improve Cache Routing, 3rd International Web Caching Workshop, June 1998
- [6] Edwin N. Johnson, "A Protocol for Network Level Caching," MIT Master's thesis, May 1998
- [7] K. Rothermel and R. Popescu-Zeletin, Eds., Mobile Agents, Lecture Notes in Comp. Sci. Series, vol. 1219, Springer, 1997.
- [8] Y. Yemini, "Network Management by Delegation," Integrated Network Management II, Krishnan and Zimmer, Eds., Elsevier, 1991
- [9] J. Vitek and C. Tschudin, Eds., Mobile Object Systems: Towards the Programmable Internet, Lecture Notes in Comp. Sci. Series, vol. 1222, Springer, 1997
- [10] M. Baldi, S. Gai, and G. P. Picco, "Exploiting Code Mobility in Decentralized and Flexible Network Management", K. Rothermel and R. Popescu-Zeletin, Eds., Mobile Agents, Lecture Notes in Comp. Sci. Series, vol 1219, pp. 13-26, Springer, 1997

ADDITIONAL READING

- [1] <http://tns.lcs.mit.edu/activeware>
- [2] <http://sds.lcs.mit.edu/activeware>
- [3] http://www.ito.arpa.mit/Solicitations/PIP_9704.html
- [4] <http://www.cc.gatech.edu/fac/Ellen.Zegura/active.html>