Data Retrieval Mechanism for Managing SNMP Enabled Networks Using ANTS Active Framework

Victória Damasceno Matos¹, Jorge Luiz de Castro e Silva², José Neuman de Souza¹

¹Universidade Federal do Ceará (UFC)
²Universidade Federal de Pernambuco (UFPE)

victoria@lia.ufc.br, jlcs@cin.ufpe.br, neuman@ufc.br

Abstract. The active networks approach has been presented as an alternative technology for solving several problems in conventional networks, mainly in the network management area. New concepts were developed by many researches for improving their efficiency and providing a better flexibility and distribution. This work describes a data retrieval mechanism test bed, including design and implementation, which can be used for managing SNMP enabled networks, based on the ANTS (Active Node Transfer System) toolkit framework developed at the Massachusetts Institute of Technology (MIT). The results issued from the experiments have shown that the chosen approach is a good alternative for data retrieving operations, regarding the network management application with respect to performance on bandwidth needs, traffic load and manager overloading.

Keywords: management network, data retrieval, and active networks, SNMP, ANTS.

1. Introduction

It’s notorious that the Internet is increasing in an exorbitant way. Hence monitoring the network has become essential for obtain a certain understanding of its performance. Many research activities have been developed on measuring and analyzing the network performance, so that the networking research communities have converged to the common understanding that a measurement infrastructure is inevitable for a optimal operation and future growth of the Internet [1, 2].

The performance management is the greatest motivator for studying data retrieval efficiently than all the others network management functional areas. It is directly related to traffic information availability, so it can determine the network load. Differently of fault management, the performance management is pro-active, involving the collect and analysis of periodical performance indicators, identifying congestions, valuating tendencies and forecasting the network future behavior. This kind of management takes care of network actual performance, including statistics parameters such that: delay, outflow, availability and number of retransmitted packets. In that case, it’s necessary dynamic access to a certain amount of network data obtained through some data collect method.

Another important concern of this work is the network management applications that might have benefices from using active network paradigm. Regarding active network
studies [3, 4, 5, 6, 7], it has been verified that one of the biggest platforms embedding this technology is the ANTS (Active Node Transfer System). That’s why the proposed data retrieval mechanism was designed and implemented in this framework.

The motivation for the proposed model came from the fact that traditional management approaches like SNMP (Simple Network Management Protocol) and CMIP (Common Management Information Protocol) generate excessive network traffic, once they have mechanisms based on the client-server model, where the manager centralizes the information and provides scheduling for execution of corrective actions, while the agent interacts with the MIB (Management Information Base) and executes the manager requests. Thus, the more complex management systems need bulk messages exchanges between the manager and the agents. Beside that these mechanisms present high delays for reporting an anomaly situation or starting corrective actions. These actions, for its time, used to be executed only after the fault occurrence, that is, they are reactive. The suggestion is to have pro-active decentralized management, aiming delays and traffic reductions.

In the proposed model, some procedures were implemented within the ANTS platform, enabling the data capture from an initialization process within an active node, that is, an active node send capsules (corresponding to traditional network packets) that execute locally the collect task. For this, these capsules have Java code that is responsible for visiting actives nodes that are being monitored and making data retrieval. That way the nodes in question will send only the expected traffic parameter results.

With this model, which provides dynamically network traffic parameters, it’s possible to assist performance management methods, mainly those that use active technologies.

Following the paper organization, the next section surveys the active networks and presents existing architectures, with emphasis in the ANTS platform, which is the framework used in the proposed retrieval mechanism. More details of this model are described in the section 3. Finally, the section 4 and 5 present the benchmarks and conclusion, respectively.

2. Active Networks

Traditional networks guarantee data sharing in the sense of their packets can be efficiently carried between connected systems. They make only the processing needed to forward the packets to destination. This kind of network is insensitive to the packets it carries and they are transferred between end systems without modification [8]. Computation role in these kinds of network is extremely limited.

For the time being, the technology has been cheaper and more functionality have been implemented within network to provide better services to the end users. In order to guarantee the creation and deployment of these new services is that emerged the active network concept.

Active networks are a new approach to network architecture that allows their users to inject customized programs into the nodes, enabling these devices to perform customized processing on the messages flowing through them [3, 8]. This approach can be applied for user or application.
This technology is supposed to resolve several problems with today’s networks such that:

- Integrating new technologies and standards into shared network infrastructure;
- Bad performance due to redundant operations at several protocol layers;
- Deployment of new services in the existing architectural model.

There are applications that will be of special interest [9] from a user viewpoint like capacity programming of their networks, making good use of active network basic features and allowing the creation of new protocols and innovated technology, which can be easily implemented.

Between the network implementations and applications that benefit themselves with active network architecture there are: the automation of rules update of a firewall; the development of web proxy schemes (which support active caches for storing and executing the programs that generate dynamics web pages); ability to provide the functionality of multicasting in an efficient, reliable, and scalable way (once active internal nodes can elegantly solve many current problems such as concentrated load retransmissions, useless retransmissions, duplication of packets and immunity to group membership changes); smart caching scheme (where the nodes should be smart enough to cache objects that nearby clients will request in the future and to coordinate with each other to avoid caching the objects that are already cached in neighbor nodes) and the efficient networks management and congestion control.

It’s of big interest for this work the benefits obtained from the active networks in network management applications. Once the network internal nodes are active, the network management leaves to be centralized and becomes decentralized, decreasing either response time or bandwidth. It’s possible to insert special code into the packets so that they can act as “first aid” in case of they finds out some problematic node. That way the affected node changes its state automatically, without waiting for a central management answer. The management policies can be changed easily by the inherent flexibility of active network technology.

The architectures are grouped according to their basic approach toward the realization of active networking [9]: active packets approach, active node approach and active packets and node approach. Most of the earlier active networks architectures follow the active packets approach. In this approach, the nodes are also active because they allow computations up to the application layer to take place, but no active code resides inside. Therefore, the reason for calling these technologies “active packets” technologies was that active code is carried by the packets either to be executed on the data of the same packet that carries the code, or to be executed in order to change the state or the behavior of the node.

In the active nodes approach, the packets do not carry the actual code, but instead they carry some identifiers or references to predefined functions that reside in the active nodes. The packets are active in the sense that they decide which functions are going to be executed on their data, and they provide the parameters for these functions. However, the actual code resides in the active nodes. The packets do not carry it. That is why these technologies are called active nodes technologies. The motivation for such architecture is that the active packets approach suffers from either performance related problems because safety and security requirements are huge, or capability related
problems because the only way to minimize the security and safety issues is by restricting the programs that are carried by packets.

It should be clear by now that active packets can carry code efficiently only when the code is relatively simple and restricted. On the other hand, active nodes can efficiently provide any code. However, this code is predefined because it should reside in the active node or at least to one node from which it can be downloaded. In the active packets and nodes approach, active packets carry actual code and other more complex code resides in active nodes. Usually, such architectures allow users to choose either the one or the other approach according to the nature of their application.

2.1. Active Node Transfer System (ANTS)

ANTS (Active Node Transfer System) is a toolkit, written in Java language, developed by the Massachusetts Institute of Technology (MIT), where new protocols are automatically deployed at both intermediate nodes and end systems by using mobile code techniques [4]. This process is made without the need of coordination or undesired interpolate between the coexistent protocols.

ANTS views the network as a distributed programming system and provides a programming language-like model for expressing new protocol in terms of operations at nodes. The tool provides the greater flexibility that accompanies a programming language and the convenience of dynamic deployment.

An ANTS-based network consists of interconnected group of nodes that execute the ANTS runtime; the nodes may be connected across the local or wide area and by point-to-point or shared medium channels. Different applications are able to introduce new protocols into the network by specifying the routines to be executed at network nodes that forward their messages. Application may customize network – either processing that is traditionally performed at end-systems or novel kinds of processing that only make sense in the context of active networks [4].

The ANTS project is based in three goals [10]:

- The nodes of network must simultaneously support a variety of network protocols providing different services.
- New protocols must be accepted by mutual agreement among interested parties, rather than requiring new protocols to be registered in a centralized manner.
- New protocols must be dynamically deployed, since it is unreasonable to take portions of the network “off-line” in order to configure nodes support new protocols.

Three key components are utilized in the architecture for obtaining the goals mentioned previously: (1) Capsules that refer to the processing to be performed on their behalf replace the packets in traditional networks. (2) Routers and end nodes are replaced by active nodes that execute capsules processing routines and maintain their associated state. (3) A code distribution mechanism ensures that processing routines are automatically and dynamically transferred to those nodes that are needed [4].

The utilization of network programming elements is made through of one model for combining forwarding routines at individual node into a pattern of behavior. This pattern defines the processing to occur across the network as whole and if model must
separate patterns of behavior from each other in the future. For that these goals have been obtained are utilized: capsule, code group and protocol.

The capsules include a reference to the forwarding routines to be used for processing at each active node. There are forwarding routines that will be found in all the nodes. Others are specific of the application and they will not reside at every node, but must be transferred to a node by the scheme of code distribution before capsules of that type can be processed the first time. The code group is a collection of related capsule types whose forwarding routines are transferred as a unit by the code distribution system. Finally, the protocol is a collection of related code groups that are treated as a single unit of protection by active node.

The active nodes have the responsibility to execute protocols within restrict environment that limits their access to shared resources. They export a set of primitives for uses by application-defined processing routines, supplying the resources shared between protocols and enforcing constraints on how these resources may be used as protocol are executed.

Given a programming infrastructure, a mechanism is needed for propagating program definitions to where they are needed. A good scheme must be efficient, adapt to changes in node connectivity, and limit its activity so that the network remains robust.

Many different mechanisms are possible. At one extreme, programs may carry within every capsule. This scheme is only suited to transferring extremely short programs when bandwidth is not at a premium. At the other extreme, programs may be pre-loaded into all nodes that may require them by an out-of-band or management channel prior to using a new protocol. These schemes are not suited to the goals of rapid and decentralized deployment.

The approach utilized is to couple the transfer of code with the transfer of data as an in-band function. This mechanism has several advantages, once it limits the code distribution only to where it is needed.

The constituted scheme transfers the code on demand and caches it to improve the performance in the common case of flows, that is, sequences of capsules that follow the same path and require the same processing. The capsule of a new kind may be injected into the network. They travel through network nodes. A lightweight protocol is used to transfer the capsule programs from one node to the next, where they are cached for future use.

A sequence of events that illustrates the operation of this demand loading protocol is listed below and shown in Figure 1:

- Capsules identify their types and the protocol to whom they belong as they travel.
- When the capsule arrives at a node, a cache of protocol code is checked. If the required code is not all present the capsule execution is suspended and load request for the missing portion based on the capsule type and protocol is sent to the node from which the capsule arrived.
- When a node receives a load request that it can respond it does so immediately. It sends load responses that contain the portion of protocol code that is implicated.
When a node receives a load response, it incorporates the code into its cache. If the required code is now present, it wakes up sleeping capsules. If the required responses are not forthcoming, sleeping capsules are discarded within further action.

3. Proposed Retrieval Mechanism

The test bed active framework is composed of peer machines with Linux operation system and ANTS package (version 2.0.2). It’s worth to say that this active network environment is completely oriented to programming Java language. The designed and implemented module is composed of some scripts that are responsible for nodes configuration that are monitored and forwarding application capsules and Java classes that are responsible for structuring capsules and data collecting process.

In other words there is interaction between the application and the SNMP agents to get MIB information within analyzed nodes. The Figure 2 shows how it works. It sends capsules with routines that must be executed in the monitored node to obtain the requested information.

3.1. Implementation of Active Collection

The first step of the implementation was to establish an ANTS active environment then an active application executes the routines desired in a node. This process of node activation is made using one of the main class of ANTS package: the
ConfigurationManager. To activate a node is necessary the association of a virtual IP and an entry to one determined active application in this node, where such configuration is passed as parameter through script (coleta.config) to the ConfigurationManager. Thus, when the capsule of a certain application arrives in a node that is active, it executes the desired routines.

Another important factor are the routes to be followed by capsules. The forwarding must be pre-established in one script called coleta.routes, which can be generated by the ANTS.

Besides that (scripts and ANTS package) the application is ready to work. It is composed essentially by three distinct classes: ColetaApplication, ColetaCapsule and ColetaReturnCapsule.

The ColetaApplication class starts the data capture, manages the sending and receiving of capsules and finishes all the process in the manager node. The ColetaCapsule creates the capsules that embed the local collection code that will be sent to the monitored nodes. That way it is possible to send new collecting capsules for these monitored nodes, characterizing the distributed model. Finally, the ColetaReturnCapsule creates the capsules that will return the results and treats the values returned for the manager node. Figure 3 shows the steps of this execution model.

3.2. Advantages of the Active Retrieval Mechanism

A comparative analysis was made between the proposed mechanism and the native model of management protocol using SNMP (Figure 4) to show the performance of both approaches.
In this environment, based on the client-server model, there is a great number of messages exchange, therefore the manager is interacting with all the managed devices to get the management information from agents MIB. This can result in decreasing performance, so the management centralization will overload the manager and the extreme messages exchanges between the manager and the agent will increase the network traffic.

The proposed retrieval mechanism in question improves the performance of getting information from MIBs, having basically two objectives: optimizing the number of management messages exchanges and decentralizing the collections.

Comparing this model with the traditional SNMP behavior: with the use of the SNMPv1 (SNMP version 1), when it was desired to get more than a value of MIB OID (Object Identifiers), the manager had to send a SNMPv1 operation (get) to each desired OID. Even with versions 2 and 3 of the SNMP (SNMPv2 and SNMPv3), where an operation exists to support a bigger number of OIDs (getbulk), it still has a limit of information that can get with the SNMP in only one message. With the proposed framework, a bigger number of values in an only capsule can be gotten, so it will be only the specification of the routines that will be executed locally and, in that way, the number of interactions between the manager and the agent will reduce, having the SNMP running only in the managed device.

The decentralization of the management process is the major advantage of this model; therefore, although SNMPv2 and SNMPv3 provide some abilities to support networks distributed management, the traditional managements are inefficient to implement real distributed management environment.

As seen previously, the traditional form of getting MIB data are centralized, therefore the manager has to capture information of all the devices that it manages. In the case of the implementation in question, an active node, that is not the manager, can send a capsule to run the collection in one another active node and an only capsule can run the operation of collection in some devices, as represented in Figure 5.

Figure 4. Traditional network framework with SNMP protocol
When the process is started, an only capsule sent to agents can acquire information of all monitored node. This is possible because an agent is able to send collection capsule to the other agents. The obtained results will return to the management station through capsules.

4. Obtained Results

Comparing the time expended in the proposed collection model with the traditional model of SNMP, using tool ucd-SNMP [11], it was evaluated that the capture of the entailed values of some OIDs, in only one node, results in a bigger time with the proposed mechanism. But when it was added some nodes to the process of collection and used the characteristic of mechanism distribution, as explained in Figure 5, the performance was improved significantly. This conclusion was arrived through the obtained average results that are presented in the Figure 6 graphic, where in scenery two nodes (manager and agent) were used and another scenery three nodes (one manager and two agents). The benchmarks ware made with and without distributed forwarding and the varied collected OIDs numbers.

A more significant profit was found when the number of OIDs to be consulted was modified. Using the scenery with three machines, concluded that the increase of the number of OIDs to be obtained has more impact in the times expenses when the environment is of the traditional SNMP. This is explained by the fact of that, in the proposed method, the manager sends only one capsule with all the OIDs to be collected; while in the traditional SNMP some PDUs (Protocol Data Unit) are sent to get data. Figure 6 shows such results.

However, it is important to say that it will happen a delay of one second, in average, due to the processing time to activate involved node. Thus, it can be verified that the model of active collection will be more efficient when the scenery of environment to be monitored is composed of some devices and the amount of variables to be collected will be raised. Therefore, despite time tax for the activation of nodes, the use of this process will be advantageous. Independently of the performance, the basic objectives of the project (to reduce management messages exchanges and to decentralize collections) had been reached.

![Figure 5. Distributed framework of data collection](image)
In the practical study it was verified that the configuration of the routes also is an excellent factor to obtain better times, therefore in an environment with many machines, is not enough to have a management distributed without evaluating where points would be advantageous to make the decentralization. It was verified that in a monitoring of few machines, the collection system must be totally distributed, or either, a monitored node to pass the collection capsule one another node to be monitored. However, when the number of nodes increases, it can have a delay in the collection with this model of decentralization. Therefore, it would be of great value the study and the implementation of the excellent solution in the establishment of the routes.

5. Conclusions

The motivations of this work are the deficiencies presented in the management traditional protocols, based in the client-server model, that are management centralized models, with extreme exchanges of messages between an agent and a manager, wasting bandwidth and getting delays in the network monitoring.

For the implementation of this new collection model, the concepts of active networks had been used, a time that the networks management is one of the applications that can be benefited with the use of this technology, since in this environment it has more flexibility, what it makes possible the decentralized management, diminishing as the reply time delay as the use of the broadband.

In the made studies, it was evidenced that one of the more important active platforms is the ANTS that is a toolkit based in Java developed for the Massachusetts Institute of Technology (MIT) dynamically to construct and to implement network protocols. Therefore, this framework was used for the development of the proposed module.

The objectives to optimize the use of the broadband and to diminish the times expenses in a collection had been reached, because is a data collection mechanism based in
distributing management and by the fact of being injected in the network only the capsules that will all indicate the starting of the local collection process, that will be through local SNMP operations, preventing the excess of messages exchange between the customer and the server. They had been made benchmarks that they had proven these advantages brought for the active data collection.

This data collection module is a contribution for that the management tools can improve its performance through characteristics of the technology of active network, as, mainly, decentralization and flexibility.

6. References


