ROUTING DOMAIN AUTOCONFIGURATION FOR MORE EFFICIENT AND RAPIDLY DEPLOYABLE MOBILE NETWORKS

K. Manousakis[i], A. McAuley, R. Morera, J. Baras[j]

{kerk@glue.umd.edu, mcauley@research.telcordia.com, raquel@research.telcordia.com, baras@isr.umd.edu}
Telcordia Technologies, Inc.
445 South Street, Morristown, NJ USA

ABSTRACT

One approach to achieving scalability in rapidly deployed dynamic networks, such as Future Combat Systems (FCS), is to automatically divide nodes into small (e.g., 30 node) interconnected IP domains and assigning each a routing protocol that best meets the domain's characteristics (Morera and McAuley, 2002). Although an attractive idea, it has not been tested.

This paper[*] presents the first realization of domain autoconfiguration through extensions to the IP Autoconfiguration Suite (IPAS) (McAuley et al., 2001; McAuley et al., 2002; Cheng et al., 2002). While IPAS already configures and reconfigures information such as interface IP address and server locations, it assumes a single domain. We describe IPAS enhancements that support the automatic creation and configuration of multiple domains and describe a prototype implementation where interfaces are dynamically assigned to run different routing protocols. Finally, we show some initial performance results that show configuration time for even large (1000 node) sparse networks in a few seconds and small bandwidth overhead (a few hundred bytes per link).

1. INTRODUCTION

FCS networks may encompass a large number (e.g., 10,000) of rapidly deployed nodes with heterogeneous characteristics and capabilities. The communication links will also have vastly different speed, range and error rate characteristics. Most networking protocols, however, are suited only to particular node and link characteristics and scale only up to a maximum number of nodes. For example, routing protocol performance quickly degrades (e.g., due to slow convergence time) if nodes are "too dynamic" or the number of nodes exceeds some maximum.

Dividing the network into independent and more homogeneous "domains," with some abstraction of intra-domain information, can help network scalability and survivability (Morera and McAuley, 2002). However, such domains must be automatically configured to meet the rapidly deployable network requirements of FCS and must adapt to network dynamics (e.g., topological changes or new mission requirements). As there has been no capability to do this within network configuration protocols, this approach has never been tried.

Here there is a practical example of how under certain circumstances army missions need domain reconfiguration to ensure network survivability. Assume a group of soldiers involved in an exploring mission forming a wireless ad-hoc network. At network deployment, all links are high quality and there is a significant amount of traffic among the nodes. Then, the single domain configuration where nodes run a unicast routing protocol works well. As the mission evolves, a small group of soldiers moves into a region with very poor radio link quality and high link failures. The unicast routing protocol is not able to cope with such network dynamics and communication between nodes becomes very difficult. With a domain autoconfiguration suite, a command can be sent to reconfigure the network and split the original routing domain in two routing domains, one still running the unicast routing protocol and the other using flooding. This way, nodes in the stable region are not affected by the unstable links in the dynamic region.

This paper describes the enhancements required by the IP Autoconfiguration Suite (IPAS) (McAuley et al., 2001; McAuley et al., 2002), used in the CECOM MOSAIC advanced technology demonstration (Cheng et al., 2002), to support routing domain autoconfiguration. Sections 2 and 3 give an overview of the current IP Autoconfiguration Suit (IPAS) and domains (Morera and McAuley, 2002). Section 4 presents our analysis of the type of domain information required for configuration and how such information can be best distributed. Section 5 describes the proposed

[*] This work was supported by the U.S. Army Research Laboratory. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

[i] Kyriakos Manousakis is pursuing a Ph.D. at the University of Maryland. This work was done during a summer internship at Telcordia.

[j] Professor Baras is the director of the Center for Satellite and Hybrid Communication Networks at the University of Maryland, College Park

extensions and modifications to IPAS for the support of multiple domains. Finally, section 6 describes the testbed and performance analysis.

2. AUTOCONFIGURATION SUIT

A complete autoconfiguration suit must encompass the following elements,

- Distribution Mechanism. Need mechanisms to robustly distribute configuration information to each interface and node in the network.
- Reporting Mechanism: Need a way to know what is in the network and its characteristics (e.g., link quality, a node's ability to be a DNS server).
- Brains. A set of rules/policies to configure and reconfigure the network. These rules can be a set of optimization algorithms that will take some metrics into account (link quality, number of nodes, traffic…).

In this section we discuss why we base our domain configuration on IPAS, describe its main components and how it encompasses all the aforementioned elements.

2.1 Why IPAS?

Rather than designing domain autoconfiguration protocols from scratch, our objective was to enhance existing solutions. However, commercial IP autoconfiguration protocols (e.g., Dynamic Host Configuration Protocol (DHCP) (Droms, 1997) and IPv6 Stateless Autoconfiguration (SA) (Thompson, 1998)) do not provide sufficient basis to build a domain autoconfiguration solution. Mainly because DHCP and SA only configure hosts (not routers) and do so only within a single subnet. Even, when SA can support multiple subnet configuration it does not provide a mechanism to IP address distribution.

The only current basis for domain autoconfiguration is to extend the IPAS (McAuley et al., 2001; McAuley et al., 2002), used in the CECOM MOSAIC Advanced Technology demonstration (Cheng et al., 2002), as it encompasses all the necessary elements of a complete autoconfiguration suit and provides complete IP configuration for hosts, routers and servers in a network. However, IPAS currently has no notion of domain or border nodes. Then, some enhancements are required in order create multiple domains.

2.2 IP Autoconfiguration Protocol Suit (IPAS)

This subsection gives an overview of the IP Autoconfiguration Suite (IPAS). More details of the IPAS components can be found in the references (McAuley et al., 2001; McAuley et al., 2002; Cheng et al., 2002).

Figure 1 shows the IPAS components that instantiate each of the main functions of a complete autoconfiguration suite. The Dynamic Configuration Distribution Protocol (**DCDP**) and Dynamic and Rapid Configuration Protocol (**DRCP**) do the configuration distribution. The Configuration Database stores the configuration and network information reported by the Update Protocol (**YAP**) . And, the Adaptive Configuration Agent (**ACA**) is the "brains" in the configuration process.

The configuration process can be pictured as a closed feedback loop. The ACA distributes new configuration through DCDP to nodes in each subnet. DRCP configures the interfaces within a subnet. Interfaces, configured by DRCP, report configuration information and nodes capabilities to the configuration server via the YAP protocol. The configuration server stores this information in the configuration database. To complete the cycle, the ACA node contacts the Configuration Database locally or remotely to get the latest configuration information. After processing this configuration information, the ACA may decide to reconfigure the network and distribute new configuration information, starting the cycle once more.

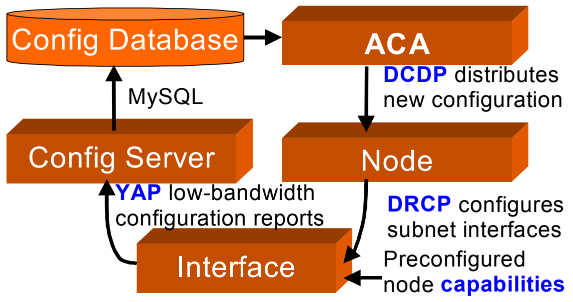
Config Database
ACA
MySQL
DCDP distributes new configuration
Config Server
Node
YAP low-bandwidth configuration reports
DRCP configures subnet interfaces
Interface
Preconfigured node capabilities

Figure 1 IPAS Components

The following paragraphs explain in more detail what are the functionalities of each of the IPAS modules and how they interoperate.

DCDP

At the heart of IPAS (see Figure 1) is the Dynamic Configuration Distribution Protocol (**DCDP**) (Morera and McAuley, 2002). DCDP is a robust, scalable, low-overhead, lightweight (minimal state) protocol designed to distribute configuration information on address-pools and other IP configuration information (e.g., DNS Server's IP address, security keys, or routing protocol). Designed for dynamic wireless battlefield, it operates without central coordination or periodic messages. Moreover, DCDP does not rely on a routing protocol to distribute information

DRCP

DCDP relies on the Dynamic and Rapid Configuration Protocol (**DRCP**) to actually configure the interfaces.

DRCP borrows heavily from DHCP, but adds features critical to roaming users. DRCP can automatically detect the need to reconfigure (e.g., due to node mobility) through periodic advertisements. In addition, DRCP allows for: a) efficient use of scarce wireless bandwidth, b) dynamic addition or deletion of address pools to support server fail over, c) message exchange without broadcast, and d) clients to be routers.

YAP

The Configuration Database Update Protocol (**YAP**) is a simple bandwidth efficient reporting mechanism for dynamic networks. YAP has three elements: 1) Clients running on every node periodically report its node's capabilities, configuration, and operational status, 2) Relays forwarding information from clients to a server, and 3) Server storing the information in a configuration database (see Figure 2). The *capabilities* say, for example: "This node can be a DNS server with priority 0" or "a YAP server with priority 3" (priority reflecting a node's willingness to perform a function). Other YAP information includes name and IP address, Rx/Tx packets, bit rate, link quality, routing table, and address pool.

ACA

The brain of IPAS is the Adaptive Configuration Agent (ACA). The ACA can even reset the network and distribute a new address pool from human input or from a predefined private address pool (e.g., 10.x.x.x). The configuration decisions are distributed to the network's nodes through the DCDP process. Through the Configuration Database (filled by YAP), ACA observes the state of the network, which allows it to initiate reconfiguration based on its rules or policies. The rules in the ACA are specific to the mission and network characteristics. Currently, ACA has a few simple and general rules, such as selecting a new DNS server if the current one does not report.

2.3 Interoperation of IPAS Modules

In each subnet there is at least one DRCP server responsible to configure interfaces. The rest of nodes in the subnet may perform as DRCP clients. The ACA runs on a dedicated node in the network. The YAP server, the ACA and the mySQL can either be located in the same node or distributed. However, for load balancing and fault taulerant reasons those entities better be distributed.

In each node, independent of its capabilities, there are three processes running:

- DCDP
- DRCP (server or client)
- YAP (client, relay or server)

Figure 2 shows how these processes communicate on a single node and how interoperate in different nodes in the network. If the node is the ACA, then its ACA process passes information to DCDP. The DCDP processes communicate with other nodes to distribute configuration information. At each DRCP server, the DCDP process passes configuration information to the DRCP process, so this can configure interfaces in the corresponding subnet. If there is a change or update in configuration information the DRCP process informs the local YAP client, which sends the information to the YAP relay for the subnet, which in turn relays the information to the YAP server. If the node is the YAP server it collects the information and stores it either locally or remotely on a configuration database. The ACA node can contact the Configuration Database locally or remotely to get the latest configuration information.

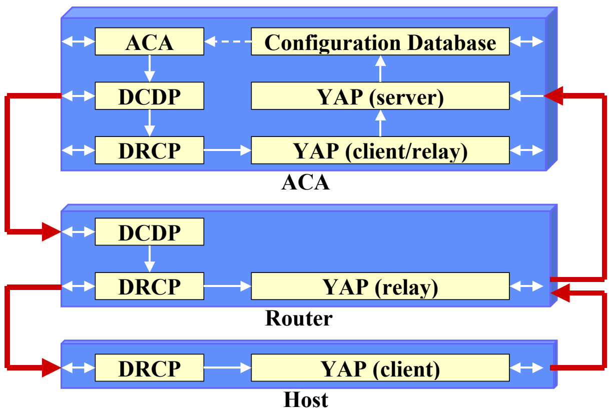

Figure 2 IPAS inter-process and inter-node communication

3. TOPOLOGICAL DOMAINS

Domains follow the architectural framework presented in (Morera and McAuley, 2002). Such framework extends the use of domains beyond mere administrative domains (autonomous systems) and focuses on the use of domains to provide scalability to the network layer protocols.

A group of topologically connected interfaces or nodes running a common networking protocol define a topological domain. We can define different types of topological domains depending on the networking protocol we focus on, i.e. configuration, routing, security, QoS and multicast domains. Each domain type is defined independently of the rest, i.e. a group of nodes that belong to the same configuration domain can belong to different routing domains.

Nodes within the same topological domain must be able to communicate without going out of the domain boundaries. Each domain runs intra-domain networking protocols independently of other domains, and border nodes allow for inter-domain communication. For example, all the interfaces in one routing domain run the same routing protocol and can reach any other interface in that routing domain without going outside the domain. Border nodes perform routing aggregation functions for the domain and export this aggregated information to other domains. Border nodes are also responsible for importing information from other domains (which may be running a different routing protocol).

A domain hierarchy is defined to allow for network and protocol scalability. Level-1 is the lowest level in the domain hierarchy and it is defined by a set of interfaces that can reach all other interfaces within that domain without going outside that domain. In an Ethernet network, for example all nodes are directly connected to each other (one IP-hop neighbors). Typically, a level-1 domain corresponds to current IP subnets. Nodes that have interfaces in multiple level-1 domains must in turn become members of a level-2 domain. The hierarchy is defined recursively; thus, nodes with interfaces in multiple level-i domains must become members of a level-(i+1) domain.

Figure 3 shows a simple example of a two domain hierarchy for an adhoc network. In ad-hoc networks such hierarchy can be dynamically created as domains are being configured. The nodes, all with a single interface (and represented by small circles with lines to their 1-hop neighbors), are grouped into four level-1 domains (“1a,” “1b,” “1c” and “1d”). Some border routers (shown with hash lines) are members of multiple level-1 domains and are also part of a single level-2 domain “2g.” The two border routers with lines at the edge of the domain 2g are also members of other level-2 domains (not shown) and are a level-3 domain (also not shown).

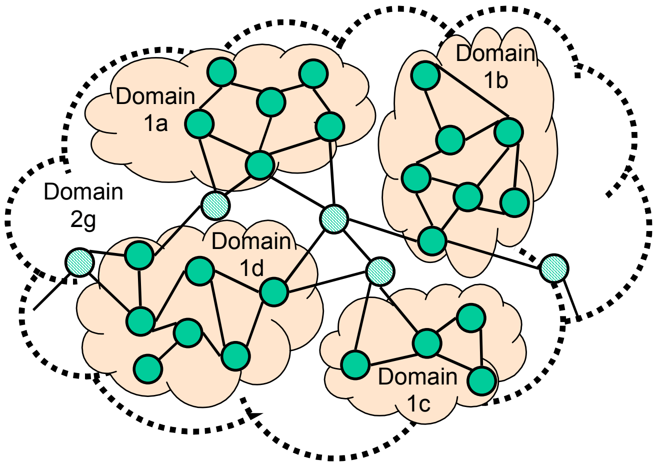

Figure 3 Example of domain hierarchy in adhoc networks

4. DOMAIN INFORMATION DISTRIBUTION

Domain configuration needs to be distributed not only at network deployment, i.e. with the initial configuration, but also as network topology and network characteristics change and the network needs to be reconfigured.

We assume all nodes in a network are initially completely unconfigured, except for some location-independent information (e.g., a node identifier and secret key). There are then many elements that must be configured (McAuley et al., 2001):

- IP address information.
- Server Location (e.g., location of DNS server).
- Networking Protocols (e.g., routing or authentication protocol).

The current IPAS provides already the mechanisms to configure these elements in a single domain. However, with multiple domains there is additional information that must be configured, such as which domains an interface is a member of and whether a node is a border node.

Moreover, nodes need to be reconfigured to maintain communication as nodes and links change. Thus, for example, if a host moves into a subnet where its IP address does not match the subnet prefix, then the node must change its IP address. Also if a DNS server is no longer available, the location of a new DNS server must be configured (McAuley et al., 2002). IPAS also provides mechanisms for network reconfiguration in a single domain. But, with multiple domains there is additional information that must be reconfigured. For example, if a node is running AODV and moves into a domain that is running TBRPF then it must change its routing protocol.

This section describes what information is needed to create and configure domains and how such information must be distributed in dynamic plug-and-play networks.

4.1 Domain Information

The following information is needed to properly configure domains,

- Domain Identifier (e.g. domain name) which must be unique.
- Domain membership information, i.e. what network elements define the domain (e.g. interfaces, subnets or nodes). Thus, each element must have a unique member identifier (UID)
- Domain configuration information, such as domain type (e.g., routing or configuration). In case of routing domains, nodes must also be informed of what type of routing protocol must run.

Domain names and UID can either be meaningless or reflect some of the domain or member characteristics (e.g., domain type and/or domain hierarchy). There are many different ways of obtaining unique domain names. For simplicity, in our experiments, we assign domain names in a random manner and we make sure that the same name has not been assigned to another existing domain. UIDs identify interfaces and are a random number chosen by the node. We ensure, however, that other mechanisms of obtaining unique names and identifiers can be explored in the future.

4.2 Domain information distribution mechanisms

Domain information is very critical network information and therefore the distribution mechanisms need to be very reliable in any kind of networks and environments to ensure that all elements in the domain receive the proper information at the right time. We considered different approaches for domain information distribution,

- Unicast to the specific nodes (or all the neighbor nodes) within that domain.
- IP multicast tree.
- Flooding.

We qualitatively assessed the performance of these three mechanisms according to four parameters: bandwidth efficiency, information distribution delay, simplicity, and reliance on the routing protocol. In terms of bandwidth efficiency, multicast is the best mechanism out of the three provided the overhead of tree maintenance is small. A given message only goes over a link once and it does not go to nodes that do not need the information (either because they are not at that level or are in a different domain). Unicast is only a little less efficient since it still limits information to those nodes that need it and may be the most efficient in dynamic networks where tree maintenance costs are higher. Flooding is the least bandwidth efficient in static networks, but becomes more attractive in dynamic networks. In terms of delay, there is little difference between the different methods. In terms of robustness, flooding is the clear winner. It does not rely on any unicast or multicast routing and it does not need any periodic messages.

Hence, we conclude that the best mechanism to distribute domain configuration information is by flooding. It is simple, it depends least on network dynamics and it is robust. The latter is a very important factor since the reason for reconfiguring domains may be the poor performance of the existing routing protocol.

5. ENHANCEMENTS ON IPAS MODULES

This section combines the requirements for domain configuration (from Section 4) with the IPAS protocol (summarized in Section 2).

5.1 DCDP, DRCP Enhancements

In IPAS, the DCDP module distributes the configuration information. We propose also using DCDP to distribute domain information. However, while the current DCDP implementation uses specific IP addresses and the underlying routing protocol to deliver the configuration information to the corresponding nodes, we propose enhancing DCDP to flood domain configuration information. Furthermore, we considered flooding to be the most robust mechanism to distribute any kind of configuration information (as we discussed in the previous section), so we changed the DCDP module to flood all configuration information. However, there is a difference in the flooding mechanism between domain configuration information and the initial configuration information. For domain configuration, information is flooded only through the configured interfaces (i.e., network interfaces that have been assigned an IP address) while in the initial configuration process, information is flooded through every active network interface, independently of its configuration status.

DRCP has also been enhanced so it processes and stores domain configuration information (i.e., to what domain each configured interface belongs to, interfaces that belong in domain D1 should run AODV).

In the enhanced IPAS, the interface between DCDP/DRCP at each local node must inform DCDP about the configured interfaces (i.e., the interfaces that DCDP will use for flooding).

5.2 Processing Domain Configuration Information

We here describe how the domain information distributed by DCDP is through the network and analyzed at every node. Domain configuration information is treated the same way as in the original IPAS implementation, i.e.

<u>Step I:</u> Configuration messages are distributed through the DCDP modules.

<u>Step II:</u> When configuration messages reach the corresponding nodes, the local DCDP module passes these messages to the local DRCP process. DRCP then takes the appropriate configuration actions based on the received message.

The ACA module generates the configuration commands that latter DCDP will distribute. In every subnet, interfaces are configured through DRCP which gets the configuration information from DCDP. The operation steps are described in the list below:

1. At the time there is a decision taken for the update of domains, ACA constructs the appropriate configuration message. Such message is passed to the local DCDP module.

2. DCDP starts the distribution of the configuration message. At every node, the local DCDP process checks whether the configuration message is referred to any of its local network interfaces.

3. If the configuration message refers to any of the local network interfaces, the local DCDP process passes the configuration message to the local DRCP process.

4. If the configuration message does not refer to any of is local interfaces, the DCDP module broadcasts the message to reach its one-hop neighbors.

5.3 New Domain Configuration Messages

In summary, domain messages are initiated in ACA from an optimization process (which we will not discuss here), then those messages are passed to the local to ACA DCDP process that starts the flooding. In the case where the message reaches to a node whose network interface is listed in the message, then this message is passed to the local DRCP process, which is responsible for the processing of the message. In order to enable this sequence of communication we had to design three groups of messages:

1. Messages from ACA to DCDP
2. Messages from DCDP to DCDP
3. Messages from DCDP to DRCP

From ACA to DCDP

The general format of the messages that are sent from ACA to its local DCDP process is:

OPCODE (1 byte)	**SID (2 bytes)**	**DOMAIN_NAME (10 bytes)**	**MSG_SPECIFIC_OPTIONS (size depends on the msg)**

The OPCODE field is related to the domain configuration action to be taken. The DOMAIN_NAME field specifies the domain for which the action will be taken and the EXTRA_INFO field is related to the action, since it provides the corresponding DRCP processes with the appropriate information for the successful processing of the domain message. The specific messages for the actions that are supported from IPAS are:

▪ *Single Domain Generation*: Every configured interface belongs to the same domain (For initialization purposes/ cover the case where we do not want to have multiple domains)

ONEDOMAIN (1 byte)	**SID (2 bytes)**	**DOMAIN_NAME (10 bytes)**

▪ *Domain Generation*: Specific network interfaces are configured to belong in the same domain

CONFIG (1 byte)	**SID (2 bytes)**	**DOMAIN_NAME (10 bytes)**	**# of INFCS (1 Byte)**	**UID (2 bytes)**	•••	**UID (2 bytes)**

100 interfaces

▪ *Domain Rename*: Modification/change of the current domain name

RENAME (1 byte)	**SID (2 bytes)**	**DOMAIN_NAME (10 bytes)**	**NEW_DOMAIN_NAME (10 bytes)**

The following message is needed to configure domains. In case of routing domains, the message has the following format:

▪ *Routing Protocol Assignment*: indicates what routing protocol to run in an existing domain.

RTPROTOCONFIG (1 byte)	**SID (2 bytes)**	**DOMAIN_NAME (10 bytes)**	**ROUTING PROTOCOL (10 bytes)**

DCDP to DCDP

The messages that DCDP passes to other DCDP processed through flooding are the same as the ones generated from the ACA module.

DCDP to DRCP

The messages that DCDP passes to DRCP are the same as the ones generated from the ACA module as we described above. DRCP upon the reception of those messages, processes them and utilizes the information contained in those messages (i.e., assigns the appropriate network interfaces to the corresponding domain, or enables the appropriate routing protocols for the corresponding network interfaces[ii]).

[ii] In a node with multiple network interfaces may run more than one routing protocols, since each interface can belong to different domain, where those domains run different routing protocols. In this case the routing signaling that is

6. PROTOTYPE AND PERFORMANCE ANALYSIS

In order to demonstrate the effectiveness of the domain configuration mechanisms and configuration information distribution we built a testbed. We then run an experiment were that consists on reconfiguring a single routing domain into two independent routing domains. After reconfiguration, each routing domain runs a different routing protocol and a border router acts as a gateway between the two domains. Nodes, however, keep their IP addresses.

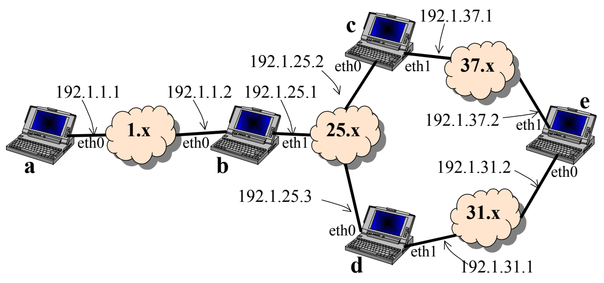

Figure 4 Domain Autoconfiguration Testbed

6.1 Testbed

The testbed is formed of five IBM thinkpad nodes (as shown in figure 4). Each node runs RedHat 7.1 LINUX has two network interfaces and is loaded with the IPAS software, i.e. every node (independently of its capabilities) runs the DRCP, DCDP and YAP modules. The first node runs the Adaptive Configuration Agent (ACA) mode, i.e. this is where all the decisions are generated and where configuration information is distributed to the DCDP process. Figure 5 shows how configuration information is distributed through the network.

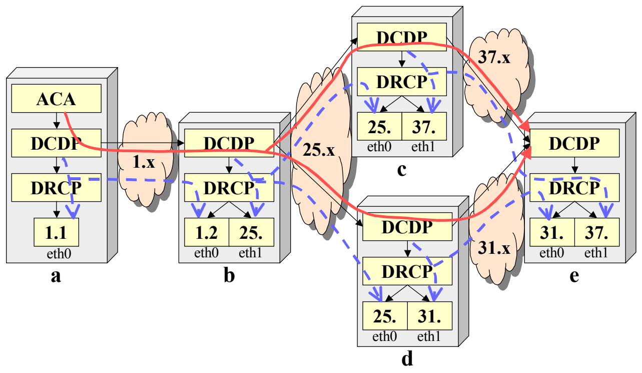

Figure 5 IPAS Message Flow

received in one interface has to be translated to the appropriate routing signaling for the other interface. This is done from the **Border Gateway** module.

Initially, interfaces are autoconfigured through the enhanced DCDP/DRCP processes. Once interfaces have an IP address assigned, we configure the routing domains and activate the appropriate routing protocols. Communication cannot start until there is a routing protocol running in each node that builds the appropriate routing tables.

First, we organize the configured nodes into a single routing domain using the ONEDOMAIN command. We then configure the routing domain using RTPROTOCONFIG, so nodes activate one of the available routing protocols. Figure 6 shows a single domain (called **d1**) running RIP.

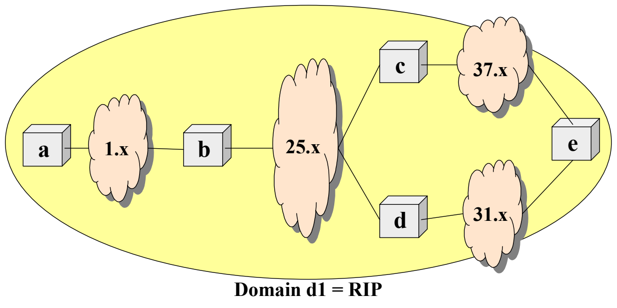

Figure 6 One domain Configuration

Second, we split domain **d1** into two routings domains (**d1** and **d2**). In order to better control the configuration, we do not run any automatic algorithms yet; rather we manually identify the interfaces (identified by their unique interface ID) that will form the new routing domain. A CONFIG message includes the domain name, the number of interfaces that are configured with that command and the interfaces ID. Figure 7 shows node **a** and the left interface in node **b** configured into a new domain **d2**. We also change the routing protocol running in the new domain to AODV with the command RTPROTOCONFIG (as we did with the original domain). Node **b** automatically configures itself as a border node between the domains with the border gateway functionalities.

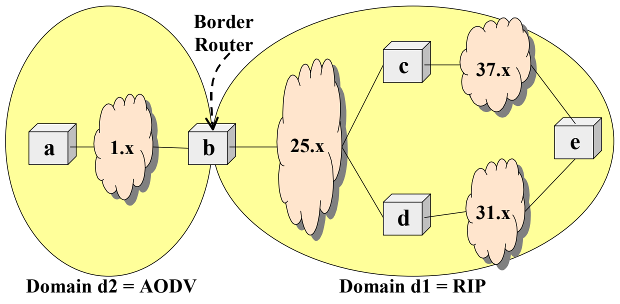

Figure 7 Two domain configuration

We also configured the network in three different domains as shown in figure 8.

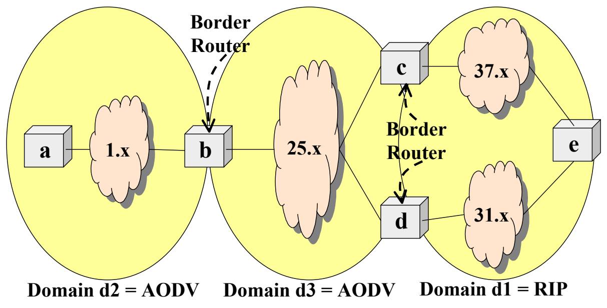

Figure 8 Three domain configuration

6.2 Measurements

We repeat the above experiment and this time we measure the time required to distribute domain information. Figure 9 shows the domain configuration time extrapolated to a thousand of nodes. The y-axis represents the time and the x-axis represents the number of configured nodes. The time here evaluated only takes into account the domain configuration information distribution, i.e. does not take into account the time it takes to restart the routing protocol. As the configuration messages are a few tens of bytes (see section 4) and information is distributed by broadcasting, networks of a thousand nodes can be configured in seconds even in sparse networks.

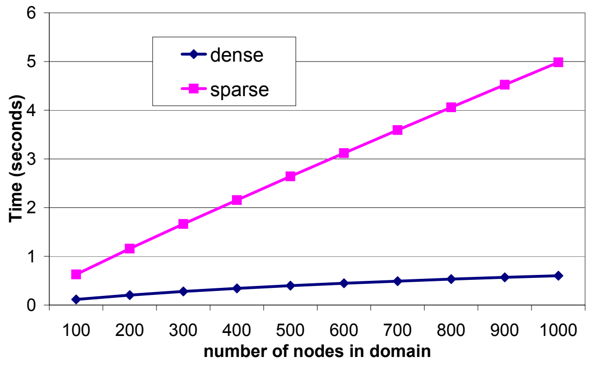

Figure 9 Domain Configuration Time

REFERENCES

Cheng T., Gurung P., Lee J., Khurana S., McAuley A., Samtani S., Wong L., Young K., Bereschinsky M., Graff C., "Adhoc Mobility Protocol Suite (AMPS) for JTRS radios," Software Defined Radio (SDR) Forum, San Diego, November 2002.

Droms R., “Dynamic Host Configuration Protocol,” RFC 2131, March 1997.

McAuley A., Misra A., Wong L., Manousakis K., “Experience with Autoconfiguring a Network with IP addresses,” IEEE Milcom, October 2001.McAuley A., et al., “Automatic Configuration and Reconfiguration in Dynamic Networks”, To appear Army Science Conference (ASC), December 2002.

Morera R., McAuley A., “Flexible Domain Configuration for More Scalable, Efficient and Robust Battlefield Networks” MILCOM, October 2002.

Thompson S., “IPv6 Stateless Address Autoconfiguration,” RFC 2462, December 1998.

CONCLUSION AND FUTURE WORK

This work represents the first part of an effort to incorporate the notion of domains into autoconfigured adhoc networks. Our implementation proves that the mechanisms can be incorportated into existing IP autoconfiguration protocols with little additional complexity, delay or bandwidth.

A lot of additional research is needed to measure how much domain autoconfiguration can improve the performance of large adhoc networks, whether in terms of robustness, scalability, throughput, security, or other measures. We must not only design decision/optimization processes, but also decide kind of information we need to collect from the network in order to reach into the optimal domain decisions quickly[i].

[i] The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or the U.S. Government