An Architecture for Context Aware Management

Catharina Candolin
Laboratory for theoretical computer science
Helsinki University of Technology
P.B. 5400, FIN-02015 HUT, Finland
catharina.candolin@hut.fi
and
Hannu H. Kari
Laboratory for theoretical computer science
Helsinki University of Technology
P.B. 5400, FIN-02015 HUT, Finland
hannu.kari@hut.fi

Abstract— **Military networks are subject to frequent changes due to hostile activities, movement of forces, weather conditions, terrain, etc. In order to function optimally, the network nodes must be able to rapidly adapt to the changes in the environment. Traditionally, this has been done by tailoring applications and protocols to exchange information. However, such solutions suffer from several disadvantages, e.g. due to their lack of generality. In this paper we present a context aware management architecture that adapts the behavior of the node to the current environment. The protocols and applications are responsible only for the tasks for which they have originally been designed, and need not be aware of the environment at all. The CAM architecture is especially suitable to mobile military ad hoc networks, which are seen as a possible future communication network solution on the battlefield.**

Keywords— **ad hoc network, core ad hoc network, context aware management, policy management**

I. Introduction

NEtworking in a military environment is difficult, since the environment is subject to frequent changes due to hostile activities, movement of forces, weather conditions, terrain, and so forth. Ad hoc networks have been proposed as a solution to battlefield networking, as they do not need to rely on a predefined infrastructure to establish and maintain communications, and they allow nodes to enter and leave the network on frequent basis and to move within the network.

Apart from the routing protocols, ad hoc networks rely on many of the same protocols as static networks do. Most of these protocols are, however, designed for connections that are fairly static. However, when the environment changes, the node must be able to adapt to the changes rapidly. If the reaction is too slow, the result may be:
1. The quality of service is bad. For example, the transmission of a video stream may be glitchy or the service cannot be provided at all.
2. Resources are wasted. For example, a video stream is transferred from a server to the ad hoc network, but the ad hoc network is unable to deliver the stream to the node.
3. An uneconomical connection is used. For example, the node has several connections to the network, but uses one that is more expensive than another.
4. The decisions are not optimal. For example, the mobility management system chooses a non-optimal access medium.

Traditionally, each application or protocol tries to adapt to the new environment independently. In some solutions, an application is specifically tailored to communicate with one other protocol layer. For example, the Real Time Protocol (RTP) [6] (and its control protocol RTCP) monitors the quality of the connection and informs the application (e.g. the multimedia video player) about degradation in quality. The application then changes the video encoding to better suit the current quality of the connection. Another example is a combination of Mobile IP [5] with a mechanism for choosing the access medium considered optimal for the applications.

The main problems with the current solutions are the following:
1. The solutions are not generic. Thus, it is hard to add context awareness to protocols and applications.
2. Protocol and application design becomes complex, as each protocol and application has to be tailored to intercommunicate with each other.
3. Old protocols and applications cannot take advantage of information provided by new protocols or applications without modification.

To overcome these problems, a new Context Aware Management (CAM) layer is added to the Internet protocol stack [4]. The purpose of CAM is to monitor the environment for changes and to adapt the behavior of the node to the current environment. The applications and protocols need not be aware of the environment at all, but rather focus on taking care of the tasks they have been designed for in the first place. For example, a routing protocol is responsible for establishing routes and forwarding packets, but need not handle issues regarding the choice of access medium. The decisions how to change the behavior of the node are made by the Policy Manager (PM). Furthermore, the architecture consists of a common database, which contains all environment related data of the node. The database is accessed via CAM. Protocols and applications communicate with each other and with the database through CAM, using a standard interface.

The rest of this paper is structured as follows: in Section II, the concept of ad hoc networking is discussed and the network architecture on which the context aware management model relies is described. In Section III, the context aware management architecture is described in further detail. Section IV describes how the context aware management architecture can cooperate with other nodes in the neighborhood by sharing information. In Section V, a case study on other ways of deploying the context aware management architecture is described. Finally, Section VI concludes the paper.

II. Background

An ad hoc network is a collection of nodes that do not need to rely on a predefined infrastructure to establish and maintain communications. Ad hoc networks are thus dynamic in nature, as they allow nodes to enter and leave the network on frequent basis. The network nodes are also able to move within the ad hoc network and between ad hoc networks. Typically, most or all nodes participate in network operations, such as routing, mobility management, and network management, depending on their capacity, location, and capabilities. This is achieved by allowing nodes to share environment information with each other.

In [2], we described a secure network architecture for military ad hoc networks. A conceptual model of the architecture is depicted in Figure 1. The picture shows how different network operations are related to each other in the node. The network interface may be any transmission medium, either wired or wireless. A node may use several network interfaces simultaneously. However, the node will typically use the network medium that is most optimal at the time. Before processing any packets arriving from the network, each packet is verified for authenticity on a node-to-node security level. If authentication fails, the packet is dropped, otherwise it is processed further by the routing engine. Routing and mobility management is performed by the routing and mobility management engines so that the former handles nano and micro mobility and the latter macro mobility. The node may use several routing protocols simultaneously, but typically uses the one that is most optimal at the time. Traffic destined for the application of the node itself is further processed by the end-to-end level security module. This module verifies the authenticity, integrity, non-repudiability etc. of the information received.

The current problem with this architecture is that the multitude of protocols and network interfaces is difficult to manage. The protocols should in principle only perform the tasks for which they have been designed. For example, a routing protocol should merely forward packets and update routing tables. It should not have to decide if it functions in an optimal fashion or whether it should shut itself down and initialize another routing protocol. The mobility management scheme should not be concerned with choosing the access medium by monitoring QoS issues, but should rather use the one that is made available to it. The context aware management scheme described in Section III shows how the ad hoc network node can cope with the variety of protocols and interfaces in a constantly changing environment.

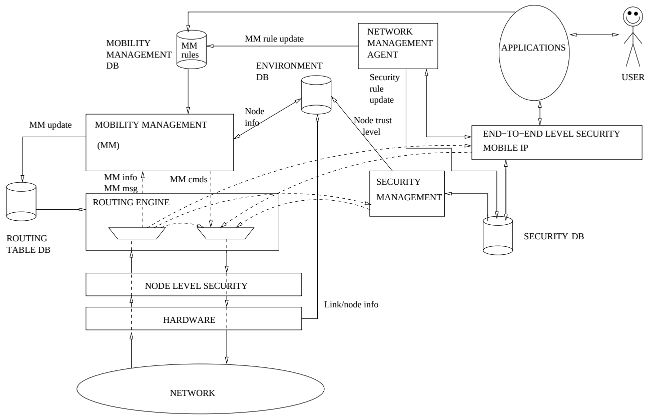

Fig. 1. A conceptual model of the network architecture

III. Context Aware Management

As the network environment changes, the node must be able to rapidly adapt to the new situation. For example, the QoS level may decrease or increase, the network topology may change as nodes enter and leave the network, the mobility rate may vary, or the hostility of the environment may change depending on the location and course of action. These changes typically require that the node switch to different access technologies, change the routing protocol, enhance the security requirements, and so forth.

To enable a node to adapt its behavior according to its current environment, a Context Aware Management (CAM) layer is added to the Internet protocol stack. The purpose of CAM is to allow modules (applications, protocols, drivers, etc.) to perform the tasks to which they are dedicated while allowing CAM to make decisions regarding the operation of the node. Furthermore, modules need not be aware of each other to take advantage of environment related information. CAM therefore functions as a common layer through which modules may communicate in a standard fashion.

The CAM architecture is depicted in Figure 2. It contains the following components:

1. The Context Aware Management (CAM) layer.
2. The Policy Manager (PM)
3. The common database (CDB)
4. The modules attached to CAM.

CAM monitors the environment for changes by providing a standard interface to all modules, through which they can communicate environment information. The information is stored in a common database. A module can provide CAM with environment information or request information from CAM. The PM makes decisions about the functionality of the node depending on the environment. It initializes the modules to be used at the time and schedules events requested by CAM.

For example, the node may be in a situation where it has

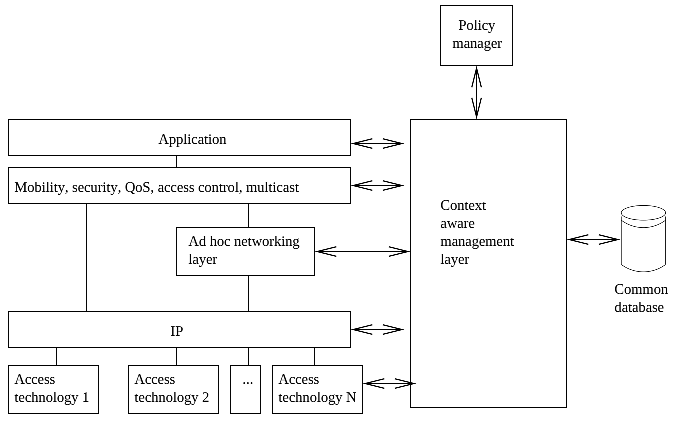

Fig. 2. The CAM architecture

very limited bandwidth available. However, an application needs to send some non-critical images to another node. Due to the limited bandwidth, the node is not able to send the images, but merely short text messages. Thus, the application will not be allowed to send the images at the current time. However, as soon as more network bandwidth is available, CAM notices it and the PM then allows the application to send the images. An event is issued to the application, allowing it to proceed. The main advantage of this is that the application has not had to deal with the reasons for why it is not allowed to send or the possible change of network medium. The application merely does what it has been designed to do in the first place.

A. *The CAM layer*

The CAM layer provides a common interface to all modules that operate in the node. All modules communicate through CAM and never directly with each other. CAM offers two interfaces; one to the modules and one to the PM.

A.1 Interfaces

CAM contains two standard interfaces: one for communication between CAM and PM, and one for communication between CAM and the modules. The common database is regarded as a module and thus uses the same interface as the other modules do.

The CAM–PM interface is defined for communication between CAM and the PM. The CAM–PM interface provides the following functions:
1. PM→CAM: register(PM, parameters). The function registers the policy manager `PM` with the given parameters to CAM.
2. PM→CAM: deregister(PM). The function deregisters the policy manager `PM` from CAM.
3. PM→CAM: set(module, parameter, value). Sets the value of the `parameter` in `module` to `value`
4. PM→CAM: get(module, parameter). Gets the value of `parameter` in `module`
5. CAM→PM: event(module, parameter, value). Requests an event to be issued to `module` when `parameter` reaches `value`

Thus, the PM can register and deregister itself from CAM. It can assign values to parameters in the different modules and read values from them. Typically the module that the PM will access the most is the CDB. The PM can also schedule events as requested by CAM.

The CAM–Module interface is defined for communication between CAM and the modules. The CAM–Module interface provides the following functions:
1. Module→CAM: register(module, parameters). Registers `module` with the given parameters to CAM.
2. Module→CAM: deregister(module). Deregisters `module` from CAM.
3. Module→CAM: set(module, parameter, value). Sets `parameter` in `module` to `value`.
4. Module→CAM: get(module, parameter). Gets the value of `parameter` in `module`.
5. CAM→Module: event(parameter, value). Issues an event to the module, assigning `parameter` to `value`.

Modules can register and deregister themselves to CAM. Registration is done e.g. when a module is added to the node or when the PM decides to initialize a sleeping module. Deregistration is done e.g. when a module is removed from a node altogether or when the PM decides to deinitialize the module. Modules can also read information from other modules and assign information to other modules. Assigning values to other modules is restricted by authorization rules. Typically, the module read from and assigned values to is the common database. There is no reason for why the modules should exchange information directly with each other. CAM can also issue events to the modules.

B. *The Policy Manager*

The PM is responsible for making the decisions regarding how the behavior of the node should be changed. The PM is aware of all modules that are loaded into the node. The PM also maintains the state information of each module. Thus, it is possible for the PM to make complicated decisions regarding the functionality of the node. For example, if the node enters an ad hoc network, the PM makes the decision regarding which routing protocol to use and with what parameters. If the node changes to another ad hoc network that uses another routing protocol, the PM may switch off the old protocol and switch on another. Another functionality provided by CAM is event handling. A module may request the PM to send a wake up signal upon the occurrence of a given event. For example, an application may request the PM to signal it once a given QoS level can be offered. This may occur when a network interface (e.g. a WLAN driver) informs the PM of a base station with sufficient signal strength. The security management module of the node may have declared that the given base station is on the list of trusted base stations and access could thus be allowed. The PM then informs the mobility management protocol to make a location update through the given base station. Once the connection is established and the required level of QoS can be offered, the PM informs the application.

The PM makes the decisions based on a set of rules. The rules can be dynamically updated during the lifetime of the node. For example, the network management system in the network may decide that the nodes need to update their policy rules, and issues each node a new set of rules. The nodes in the network then perform the update. The policy can also be changed by the user of the node. The PM of the nodes in the network can also communicate with each other and exchange environment information and even policy rules (under some circumstances).

C. Common database

The CDB contains all environment related information that is of interest to several modules. Such information may be the level of QoS, security related information, such as cryptographic material and trust levels of other nodes, and so forth.

Access to the CDB is managed through CAM. Each module is given a set of authorities as to which information in the CDB it is allowed access to and for what purposes. Typically, a module will be allowed to read a large variety of entries and to update its own entries. The PM can access and modify all entries in the CDB.

The information in the CDB is always related to a module, although several modules may access and even modify the information (depending on their authority). The module is said to be the owner of the information. Information that is not owned by any module is owned by the PM. The owner of information may change, e.g. if the routing protocol owns certain information and PM switches the protocol to another, the new protocol will inherit the information.

D. Modules

The modules are the protocols, applications, device drivers, and other pieces of software that communicate with each other and the PM via CAM. Modules are organized in a hierarchical fashion so that e.g. all network modules are organized under the category "access devices" etc. Thus, the PM is able to recognize new modules without modification. If a new category is added, the PM will be updated accordingly.

When a module registers itself to CAM, it may (if CAM requires it) provide CAM with some proof of statement signed by a trusted authority stating that the module is trustworthy, i.e. that it does what it claims to do and nothing else. If the module is not considered trustworthy, it will not be loaded into CAM.

If registration succeeds, the module will be assigned a set of authorities to CAM. The authorities state which information the module is allowed to read from the CDB and which information it is allowed to modify. Typically, the module will be allowed access to the information that is needed for it to perform its tasks as well as the privilege to update information related to itself.

A module may deregister itself from CAM e.g. if the PM decides to deinitialize it. This may be because the user has made the request to change the module, the network management system has ordered a change, or CAM has detected a change in the environment that calls for the change of modules.

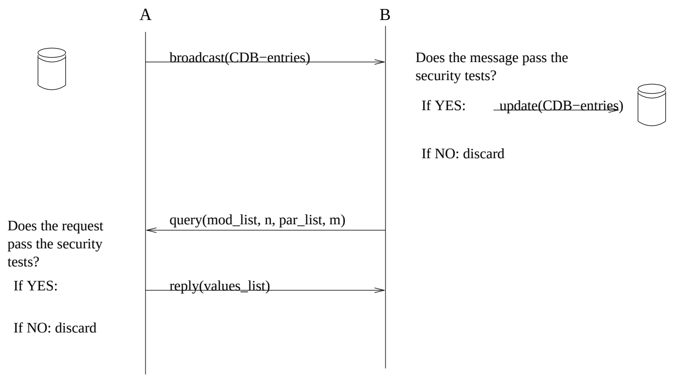
A
B
broadcast(CDB−entries)
Does the message pass the security tests?
If YES:
update(CDB−entries)
If NO: discard
Does the request pass the security tests?
query(mod_list, n, par_list, m)
If YES:
reply(values_list)
If NO: discard

Fig. 3. A C2C protocol

IV. Ad hoc network environment

One of the characteristics of ad hoc networks is that nodes are able to share information with each other. A node can retrieve network information in three ways: by own experience (as expressed by the modules), by local experience (i.e. by sharing information with other nodes in the neighborhood), or by global information distribution (e.g. from the network management system).

Each network node that deploys the CAM architecture can use it to exchange environment information. CAM contains a special CAM–to–CAM (C2C) module that is dedicated to the task of collecting and distributing information from the environment. When the C2C module receives information from another node, it first verifies the authenticity and authority of the information before passing it on to CAM. CAM does not handle this information any differently than it does information from other modules; it merely verifies the authority of the C2C module. In such a way, network information is transparent to the rest of the node. Furthermore, if another node requests information, the C2C module first verifies that the other node is authorized to request such information before retrieving it from the CDB, again presenting its own credentials to CAM. The C2C module may also distribute information to the network by broadcasting it.

A. CAM–to–CAM protocol

The C2C modules in the various nodes need a protocol through which they communicate. An example protocol is depicted in Figure 3. The protocol is very similar to the protocol used for distributing incomplete trust presented in [3]. The concept of incomplete trust was presented in [2].

The protocol provides a function *broadcast(CDB-entries)*, which broadcasts a list of entries from the CDB. The function *query(mod_list, n, par_list, m)* allows the node to query another node for a list of m parameters from a list of n modules. The function *reply(values_list)* returns the values of the parameters of the modules. If a parameter was not available, the value is set to *null*.

The *update()* procedure in the picture updates the CDB according to some predefined set of rules. The C2C module may need to scale and normalize the values received from other nodes to suit its own sense of the environment. Also, the information received from other nodes may not be given as much credibility as information collected by oneself; thus, the effect of the received information should not be as large.

V. Case study: Core ad hoc networking

Ad hoc networks can be interconnected with fixed networks in order to provide global network connectivity to ad hoc network nodes. In such a case, the ad hoc network functions as a (temporary) access network. However, also the core network is vulnerable to destruction. In [1], the concept of core ad hoc networking is introduced as a solution to the destruction of core networks. The main idea is to dynamically assign any available node from anywhere in the network to handle the tasks of the destroyed node in the core network until the network can be rebuilt. Thus, the core network is reestablished in an ad hoc fashion. However, the core ad hoc network does probably not have the same capacity as the original network, nor is it able to provide the same level of QoS. It may also have a limited lifetime if the node operates on batteries. The purpose of core ad hoc networks is to allow at least the most crucial network traffic to be transmitted in the network, as it is better than having the connections completely down.

If the core network is partially or completely destroyed, a CAM enabled node can be reassigned the tasks of a core network node by updating the PM. The new PM deregisters the modules that are no longer needed and registers the ones that are required for the new task. Also the contents of the CDB are updated. The node then starts functioning as its predecessor node. However, the ad hoc network node may not have the same capacity and resources as the previous node. Security issues are mainly related to authorization of nodes to perform network operations in the core network. In [1], certificates are used to prove authority of the node. The authority has been assigned to the node beforehand, stating that in the case of a crisis, the node is allowed to perform a given set of tasks. Should the crisis occur, the node will hook up to the core network by proving its authority. Furthermore, the node must itself verify that the request to be assigned to a new task comes from a valid source. The trust relationship between the source and the node is typically established beforehand, so normal authentication or authorization procedures can be deployed in this case.

VI. Conclusion

Networking in a military environment is a difficult task, since the network is subject to frequent changes which affect the nodes in various ways. In order to be able to function optimally, network nodes must be able to rapidly react to the changes in the environment. Traditionally, each protocol and application has tried to adapt the environment independently of each other. At most, one application has been especially tailored to communicate with a given protocol layer. The problem with such solutions is that they are not generic, application and protocol design becomes complex, and old protocols and applications cannot take advantage of information provided by new protocols.

Future military networks are likely to be mobile ad hoc networks with varying size, mobility rate, and functionality. Ad hoc networks may have a large variety of protocols to choose from depending on the environment. In order to manage the protocols to be used at each given time, a context aware management (CAM) layer is added to the IPv6 protocol stack. The purpose of CAM is to monitor the environment for changes and to adapt the behavior of the node accordingly. The protocols and applications need only be concerned with the tasks they have been designed for in the first place. The policy manager makes the decisions regarding the change of behavior according to a set of rules, which may be dynamically updated during the lifetime of the node.

The CAM enabled nodes may share environment information with each other. A special CAM–to–CAM module collects information from the environment and distributes information to the environment.

The CAM architecture can also be deployed for core ad hoc networks in case the core infrastructure has been destroyed. In such a case the node is assigned a new PM (or a new set of rules), which initializes the modules needed for the new assignment and deinitializes those that will no longer be needed.

Acknowledgments

This research was done in cooperation between the Helsinki University of Technology and the National Defence College in Finland, and was funded by the Finnish Defence Forces.

References

[1] Catharina Candolin and Hannu Kari. Dynamic management of core ad hoc networks. In *Proceedings of InfoWarCon 2002*, Perth, Australia, November 2002.

[2] Catharina Candolin and Hannu Kari. A security architecture for wireless ad hoc networks. In *Proceedings of IEEE Milcom 2002*, Anaheim, California, USA, October 2002.

[3] Catharina Candolin and Hannu Kari. Distributing incomplete trust in wireless ad hoc networks. In *Proceedings of IEEE Southeastcon 2003*, Ocho Rios, Jamaica, April 2003.

[4] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. Request For Comments 2460, IETF, December 1998.

[5] D.B. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. IETF Internet-Draft draft-ietf-mobileip-ipv6-21.txt, February 2002.

[6] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. IETF Request For Comments 1889, January 1996.