Context-Driven Self-Configuration of Mobile Ad Hoc Networks

Apostolos Malatras and George Pavlou

Centre for Communications Systems Research, Department of Electronic Engineering,
University of Surrey, UK
{a.malatras, g.pavlou}@surrey.ac.uk

Abstract. We present the design and implementation of a working prototype system that enables self-configuration in mobile ad hoc networks (MANETs) by exploiting context awareness and cross-layer design principles. The driving force behind the proposed system is to allow for self-configuration of MANETs by enabling them to be adaptive to varying conditions. Emphasis is placed on describing the requirements and specifications of the supporting platform's functionality. We propose the distributed management of the MANET through a proactively constructed body of nodes in order to cope with the inherently dynamic nature of MANETs. We present our work on deploying the designed system on our experimental MANET testbed and provide results of its performance based on extended testing.

1 Introduction

The concept of mobile ad hoc networks (MANETs) has brought a new paradigm in communication networks and acts as an enabler for pervasive computing and communication environments. In ad hoc networks, the mobile nodes (MNs) are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Conventional wireless networks require some form of fixed network infrastructure (i.e. the core network) and centralized administration for their operation. In contrast, since MANETs are self-creating, individual MNs are responsible for dynamically discovering other nodes they can communicate with. This way of dynamically creating a network often requires the ability to rapidly create, deploy and manage services and protocols in response to user demands and surrounding conditions in an equally dynamic manner.

We assert that this highly dynamic environment can benefit from the emerging context-driven autonomic communications paradigm. There has been no proper previous research on deploying autonomic communication solutions in MANETs, but such aspect is important due to their inherent nature. As such, autonomic communication principles can assist in the self-management of MANETs and enable network self-configuration and optimization by utilizing context information. The latter can be used to establish the need for automatic changes (self-configuration) in accordance to high-level pre-existing rules. Context-information can be used to trigger cross-layer changes (network and application configurations) according to predefined rules, leading to autonomic decision-making.

This paper provides conceptual and practical design, implementation and deployment issues regarding a middleware platform used for the self-configuration of MANETs. The structure of the paper is as follows. After this brief introduction, Section 2 reviews basic autonomic communication and computing principles, including pointers to related work.

Section 3 gives an overview of the proposed system's design and architecture providing justification for our choices. Details on the implementation of the platform and its deployment on our experimental MANET testbed is the subject of Section 4, where the results of our practical experimentation are also presented. Finally, Section 5 concludes the paper and discusses future research directions.

2 Autonomic Communications Principles and Related Work

Autonomic computing emerged as an initiative by IBM and has generated a very active research stream bringing together interdisciplinary domains. Autonomic computing refers to the self-managed operation of computing systems and networks, without the need for administrators but with high-level objectives dictating the system's functionality. The IBM autonomic computing blueprint [1] defines four distinct concepts behind autonomy, namely self-configuration, self-optimization, self-healing and self-protection [2]. The building block of all autonomic solutions is an autonomic element. This refers to the collection of one or more managed elements that are handled by an autonomic manager. The latter monitors the state of the elements, analyzes it and acting upon high-level objectives (typically defined as policies) imposes the execution of configuration changes on the managed elements. This process is repetitive [2], [3].

Most autonomic computing platforms are targeted to systems with sufficient resources that are relatively stable [1], [4], and [5]. The application of autonomic principles on MANETs has not been adequately researched. In [10] we presented our initial approach and results on self-configuring and optimizing MANETs. In [6] a policy-based network management system for MANETs is proposed but the hierarchical approach adopted assumes the existence of several "thick" nodes in the network, which may not always be the case.

Programmability is a very important aspect of autonomic systems, especially in ad hoc networks given the multitude of potential solutions for routing, quality of service support and other application services. Programmability can be achieved through a variety of means. Active control packets may carry code to be evaluated in routers and this approach has been used for active routing in ad hoc networks [7]. Mobile agents may be used in full mobility scenarios, carrying code and state to manipulate different MNs, or in a constrained mobility mode [8] as a more flexible means for the management by delegation approach [9]; in the latter, code is uploaded and executed in MNs through "elastic management agents", augmenting the node functionality. Programmability is also possible through the provision of suitable management interfaces that allow code to be uploaded to MNs and activated in a controllable fashion. In our recent work [11] we proposed a programmable middleware capable of dynamically deploying services and protocols in ad hoc networks.

3 System design and Architecture

We propose the deployment of a lightweight, context-aware middleware platform on every MN of a MANET and a distributed management approach based on the existence of

an adaptive set of nodes called Management Body (MB). The middleware platform is responsible for monitoring the individual MN context individually and the context of the MANET as a whole. Context information is handled locally at each MN and aggregated information is passed to the management body of the MANET. The latter reaches management decisions based on this aggregated context and in accordance with predefined rules. The corresponding configuration changes are autonomously deployed on the MNs through software plugins that carry the desired functionality.

3.1 A Hybrid Approach to MANET Management

There exist two diverse approaches regarding the management approach to be deployed in a MANET. In the hierarchical approach the MANET is grouped into clusters, each electing a local leader or cluster head (CH). The CHs act in cooperation and elect a global leader or network head (NH) that is responsible for deciding on key management issues. This approach bears similarities to the one undertaken by routing protocols such as OSPF and scales well, limiting the MN interactions within a cluster or among CHs. Moreover, it allows operation in a controlled distributed fashion, where decisions are taken not only by the NH but through cooperation and "voting" among the CHs. A diametrically different approach is a fully distributed one, in which all the nodes are deemed as equal and determine collectively any management decisions to be taken. This approach requires more complex cooperation protocols and may not scale for large networks with many MNs. On the other hand the hierarchical approach suffers from the existence of single points of failure, i.e. the CHs. In case a CH leaves the MANET or moves to a different location (and thus changes cluster), the clustering process will have to be re-initiated, an option not suitable for dynamically formed MANETs.

We chose to use a hybrid approach for our management scheme. Our approach resembles the hierarchical approach by dividing the MANET in clusters; a collaborative Management Body of MNs replaces the CH. The MB has collectively the functionality of the CH but does not suffer from single node movements as these are mitigated from interactions with the other MNs forming the MB. In a similar fashion, a collaborative body comprising selected nodes from the management body replaces the network head (Figure 1). The management decisions are taken collaboratively by the MNs assigned to the management body. Our scheme is inspired from the formation of virtual backbones in MANET routing protocols and service provisioning. The idea of using a virtual backbone to serve as a management entity in a MANET is not new. There have been several approaches in the literature that have considered similar schemes [12], [13], [14].

We chose this hybrid approach due to the fact that neither of the existing approaches suits the MANET features completely. The hierarchical approach does not perform well when node mobility is involved and is thus applicable to longer-term, relatively stable MANETs. In contrast, the fully distributed approach is very demanding as far as message exchanges are concerned and can be applied to small MANETs with few nodes. The combined approach we chose has the following benefits. The management decisions are devised by a number of nodes in the MANET and not by a single one. This distributes the load across the MANET, which is necessary for both resource conservation and reliability

& robustness reasons (i.e. avoiding single points of failure). The hierarchical features of this scheme allow for the deployment of a uniform management approach over the MANET as desired. The MBs are constructed so as to be relatively stable, while there is support for nodes leaving the MB. The MB is reconstructed only if a significant amount of MNs that comprise it leave. This ensures the avoidance of dangerous situations, with any node potentially triggering the MB formation process unnecessarily. We realise the overhead imposed on the MANET from the cooperative management architecture but we consider this a fair trade-off given the robustness achieved.

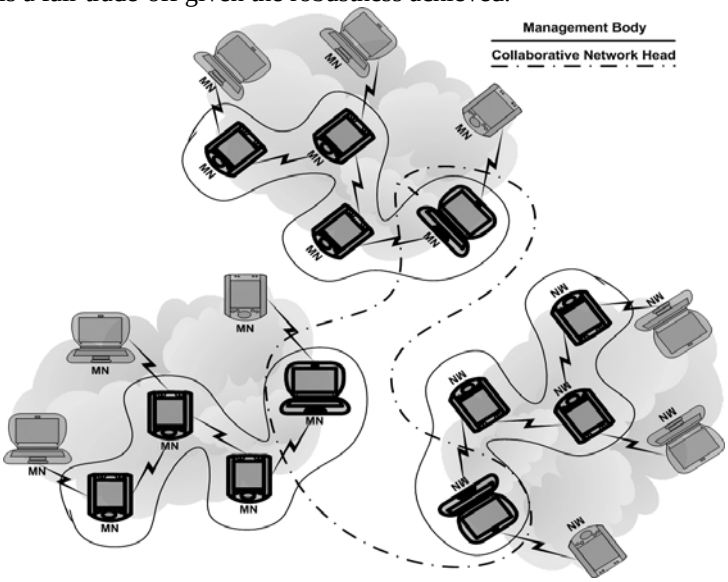


Fig. 1 Hybrid approach to MANET management

The virtual backbone used in MANETs is usually constructed as the Minimum Connected Dominating Set (MCDS) of the MANET graph. Unfortunately, the construction of an MCDS for a connected graph is an NP-Complete problem. There are two ways to face this problem, namely using an approximation algorithm or making use of a heuristic to reduce the problem into one solved in polynomial time. We chose to undertake the heuristic approach when creating the MB. Apart from that, and in favour of simplicity and timeliness, we opted towards establishing any CDS and not the minimum one. We use two heuristics to discover the CDS, the computational capabilities of the MNs (the most resourceful nodes) and their prospective, relative location stability (the nodes that are less likely to affect the network topology and thus do not lead to frequent MB re-formations).
The nodes that will be part of the MB should therefore have sufficient resources to handle the additional requirements, such as communicating with other MB-members to reach to management decisions. The nodes forming the MB are collectively the set of nodes with the highest computational resources in the MANET. Every node is calculating a value that

denotes its capability to become a member of the MB. The value of this property is then used in the selection process for the dominating set.

Our proposed capability function (CF) exploits the following attributes: memory requirements (MEM), processing power (PP), battery power (BP), mobility ratio (MR) and current load (CL). These 5 variables need to be combined in a single equation, the Capability Function (CF). MEM, PP and BP are obviously proportional to CF while MR and CL are inversely proportional. By assigning weights to these variables in accordance to their significance, we have the initial CF equation (1).

$$CF\ (x) = \frac{(w_1 \times MEM\ (x)) \times (w_2 \times PP\ (x)) \times (w_3 \times BP\ (x))}{(w_4 \times MR\ (x)) \times (w_5 \times CL\ (x))} \quad (1)$$

where, $\sum_{i=1}^{5} w_i = 1$, and x is the MN.

The main requirement for the CF is to lead to comparable results among MNs. For this reason the various attributes must be demoted in common range values. Space limitations do not allow us to delve into more details on how to achieve this. Equation 1 is used to derive a value for every MN that is proportional to its capability of being part of the MB.

Obviously, one should not expect the MANET topology to be known. Distributed approaches to construct the MB are thus adopted. The distributed construction of CDS has been intensely researched [12], [13], and [14]. We decided to take a similar approach. Details of the algorithm we have used to derive the CDS of the MANET are not presented due to space limitations. Our approach is based on building a relatively stable CDS with the "thickest" nodes according to the CF mentioned, but also takes into consideration the need for maintenance of the CDS due to the inherently unstable MANET nature.

3.2 Context Management

Autonomic communications solutions currently available have focused on monitoring device specific characteristics and network conditions in order to infer configuration adjustments on the devices or the network as whole. We differentiate our approach by extending the sensed environment to also consider user-specific information (i.e. user profiles and user explicit information) that can have an effect on the underlying network, as well as physical environment attributes with the same property (i.e. device location and vicinity information). Cross-layer context gathering is the basis of our middleware platform that exploits this information in order to allow for MANET self-configuration. The collection of context from the surroundings of the mobile nodes is handled by a series of interfaces that communicate with the available sensors, constituting the monitoring component of our platform. We consider the term context in a generic fashion, incorporating both computational and physical resources.

Each MN is responsible for collecting its own context information and processing it to higher-level context information that has an impact on the management plane of the MANET. For example a MN might collect its current location and monitor this through a GPS receiver installed on it, but this information is not useful for the MB. Useful information for the MB would be the mobility prediction for each MN, since having this

can be used for proactive configuration changes, as it will be shown in Section 4. Other higher-level context information can refer to QoS requirements, security requirements and prospective network load. This set of elaborate context information is in effect aggregated from simpler context information. The advantages of this approach are obvious. By aggregating the context information available to a MN to a set of "advanced" contexts that are passed to the MB, less control load is imposed on the MANET in terms of traffic. It also distributes the processing and storage load of handling all the context information among the MNs of the MANET. The alternative would be to pass all this information to the MB, which would then be responsible for processing it, storing it and infer configuration changes based on it. The set of advanced MN contexts that are passed around from MNs to the MB are predefined and their processing occurs using the functionality of our middleware platform as described later.

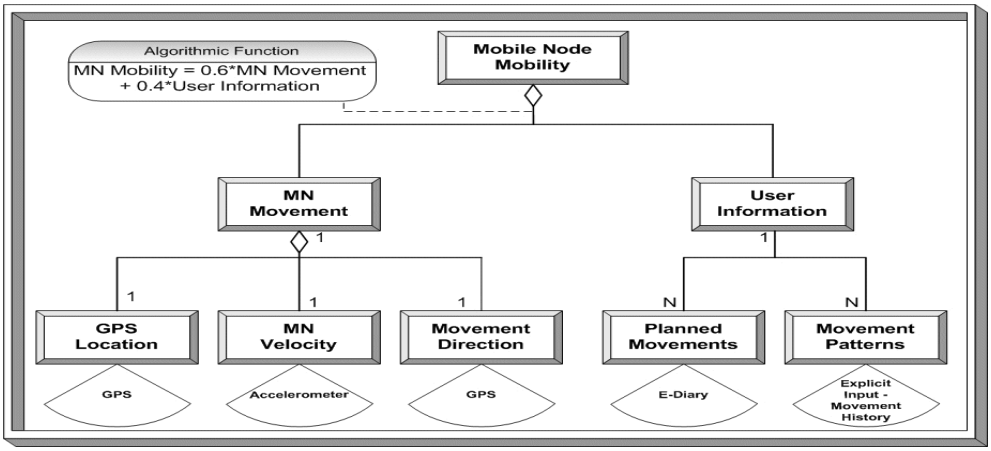


Fig. 2 Mobile node mobility context as derived from simpler contexts

Figure 2 presents an example of how the aggregated context of MN mobility can be derived from simpler contexts collected from device sensors. The analysis of elaborate contexts to simpler ones is based on the sensors used, while it should also be noted that semantic metadata information and algorithmic functions describe the way this analysis occurs in a human-understandable and a formal way respectively (e.g. in the example of Figure 2, the MN mobility is more dependent on the movement metrics rather than the user specific information since we deem the former as more credible). We represent the context using an XML-based model that takes all this information into account and allows hits lightweight processing, specific details though lie outside the scope of this paper.

3.3 MANET self-configuration

The proposed middleware platform builds on the aggregated context information that is collected from all nodes to reach to management decisions for the MANET as a whole.

These decisions are then implemented as (re-) configuration changes. Only this context information is transported across the MANET, limiting thus the traffic requirements. It also relieves the MB from a series of resource- and time-consuming processing operations, which are handled individually by every node, distributing thus the processing load. We have already mentioned that the set of aggregated context information is prespecified. The same stands for the rules that are used to establish the need for configuration changes in the MANET. The MNs forming the MB of the MANET know these rules in the form of policies. When certain preconditions are met, the rules are activated and the corresponding configuration changes are deployed on the MNs. One such example that will be elaborated in the next section is monitoring MN mobility. When the relative mobility of the MNs is changing, it might be beneficial to change the routing protocol used in the MANET. These rules in our platform are currently static and predefined. We are working towards a more dynamic and adaptive scheme based on higher-level policies, so as to increase the degree of autonomy of our system.

The configuration changes are deployed on the MANET through software plugins that carry the corresponding functionality. These plugins can be any software module, from a simple set of commands, e.g. a script, to complex applications, as long as they conform to the defined interface. All plugins should conform to standard interfaces regarding activation, deactivation and reconfiguration.

One question that arises is how the MB members collaboratively monitor and act upon the aggregated context of all MNs. For each aggregated context there is a function used to calculate its value as far as the related rule is concerned. Every MB member calculates this value collectively for the MNs it dominates and floods this information within the MB. At the end of this process every member of the MB will have a MANET-wide understanding of the rule-specific value for every aggregated context. In the previous example, relative mobility is the rule variable for routing protocol selection. Every MB member calculates its relative mobility to that of the MNs it dominates, floods this information to the rest of the MB members and receives relevant information from them. The new values it receives are used to update its relative mobility so as to include those of the rest of the nodes in the MANET.

This MANET-wide value for every aggregated context is compared against the rules in the MB nodes to establish if the need for a configuration change occurs. If so, then the appropriate action is passed from every MB member to the nodes it manages through a particular plugin. The fact that all MB members have the same values for the context and the same predefined rules ensure that the same action, if any, will be employed on the MANET, achieving a uniform self-configuration scheme.

3.4 Middleware Architecture

Figure 3 depicts the proposed system's architecture from a high-level perspective. This middleware platform is installed on every MN of the MANET, empowering it with the necessary functionality. As it will be seen at the experimentation phase, the architecture proposed is relatively lightweight. We will describe the platform and provide justification for our design choices regarding monitoring, context handling and self-configuration.

Context Monitoring

The sensed environment is accessed by means of sensors. These sensors are diverse in the way they provide the sensed information to whoever needs it. We designed a generic interface for that purpose, the Sensor Communication Interface (SCI), to which all communication protocols with the sensors conform. Every device is equipped with the SCIs for the sensors it carries and we consider them supplied as software modules bundled with the sensors. Realizing that a device might require accessing a sensor for the first time (i.e. a new positioning device) and does not have the particular SCI, we have implemented the SCI Manager. This is responsible for advertising the SCIs the device holds and discovering and retrieving SCIs from other MNs by communicating with their respective SCI Managers.

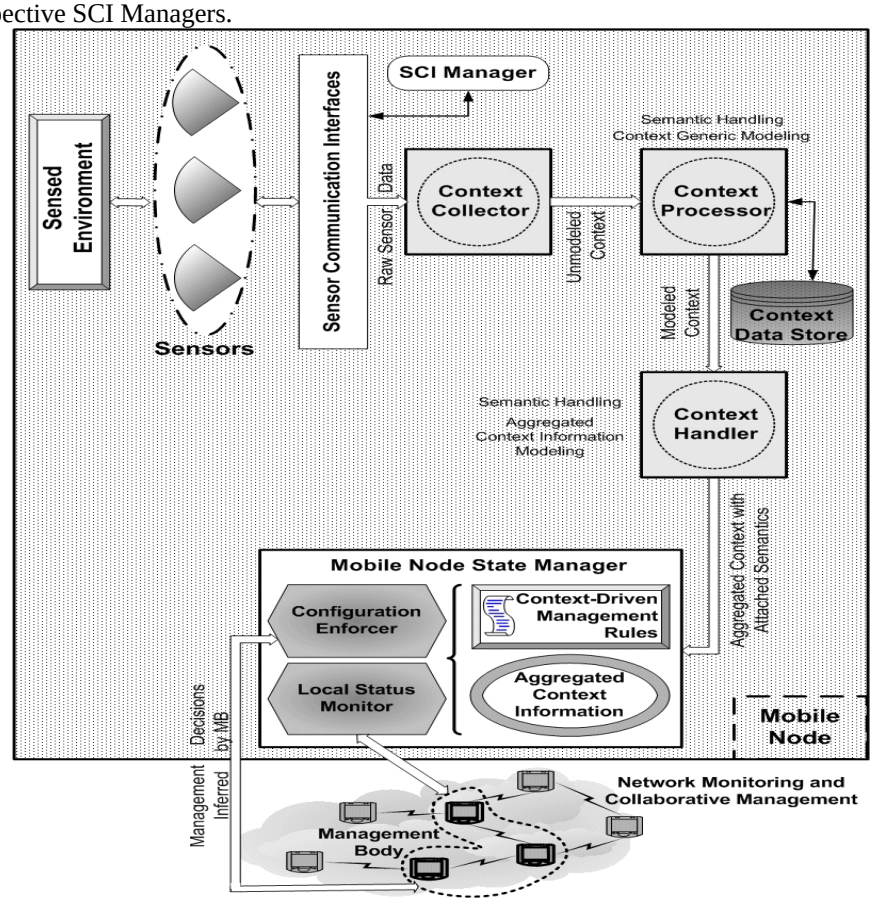


Fig. 3 Higher-level context-aware middleware architecture

Sensors do not produce context information but raw data that has to be translated into

meaningful information i.e. context. For this reason semantics regarding the data the sensors produce are included in the various SCIs so that the raw data gains some semantic meaning before it is passed to the Context Processor. The Context Collector is responsible for this task. Another task that this module is in charge of is the pruning of the abundant context information. Sensors produce a plethora of data that are not all useful. For example GPS receivers inform for every single location change, even in the scale of some meters. This amount of detail might not be needed to be collected. The Context Collector retains custom filters for each context collected that states which changes in values are deemed significant to be stored and which should be discarded.

Context Handling

The Context Processor and the Context Handler are the two modules that collectively manage locally the context information for a MN. The former is responsible for modelling the primitive context information collected from the sensors to the generic context model we have devised. Semantic information is tagged to the context in order to allow for semantic operations to be performed. The Context Collector comprises 3 entities, namely the Processing Interfaces, the Context Modeller and the Semantic Handler. The Processing Interfaces entity is used to provide different interfaces for the handling of various data types provided by sensors. One sensor might for example produce binary data and another scalar. This entity provides the generic feature for the platform to be able to respond to every possible input. The Context Modeller then is instantiated with its main activity being the translation of the simple data to the model representation proposed. The Semantic Handler enriches the semantics of the context, with metadata more specific to the uses of the platform. The sensors provide some metadata about their collected data to give an understanding of what they are monitoring. For example a GPS might yield that it is collecting MN location through a “location” value. The Semantic Handler builds on this and provides more semantics like “latitude-longitude/positioning” etc. The purpose of this is to ensure that the platform is not explicitly bundled with sensors, i.e. the “location” metadata but it is rather bundled with the general notion described by more than one words. The Context Processor stores context information in the local data store created for this reason.

The Context Handler is responsible for the task described earlier: collecting simple contexts and aggregating them to higher-lever contexts that are going to be sent to the MB. To do that in a generic fashion it exploits Context Handlers and Aggregated Context Modelling. These two entities collaborate with the Semantic Handling entity to infer useful knowledge on the aggregated context. The modelling of this higher-level context is based on predefined models that are hard-coded on the platform. The platform is open enough though to support new aggregated context models that may be required from the MB. The MB might decide for example upon using a context of MN QoS requirements. The MNs are not aware of the model to be followed to infer this context from simpler contexts. The MB members then transfer the model properties to the MNs and acting upon it the MNs respond to the MB with the desired QoS requirements context.

Self-Configuration

We consider that the functionality the MNs, regardless of the heterogeneity of the

available platforms, is manipulated and altered through software plugins. For instance, a routing protocol used by mobile nodes, is as far as our platform is concerned a loadable plugin that has open interfaces to allow its activation, de-activation or reconfiguration according to management demands. The self-configuration aspects of our context-driven middleware platform are thus implemented through the use of these software plugins that can be implemented simple scripts or Java, C/C++ or any other programming language objects in our experimental prototype.

Self-configuration is handled through the MN State Manager module. The main responsibility of this module is to collect and advertise the aggregated context information to the MB. Communication with the MB (through XML-RPC as will be elaborated later) is handled by the State Manager, as is communication with other MNs. Hard-coded into this module are the general Context-Driven Management Rules that are used by the MB to examine if necessary conditions are met and configuration changes are necessary. The Local Status Monitor has the obvious functionality of retaining and making available the information on the current local status of a MN. The Configuration Enforcer receives "orders" from the MB regarding configuration changes through software plugins. When such "orders" are given, the Configuration Enforcer imposes them on the platform by acquiring the required plugin if it does not have it and activating it.

The plugins are considered to be owned by at least some nodes of the MANET, since we cannot consider them being generated at runtime. For example, if the plugin is a routing protocol like the case study in Section 4, this must exist in some of the MANET nodes. The nodes that have the required plugin are informed by the MB to distribute it within the MANET by means of efficient flooding to their neighbours and so forth. The flooding is efficient in two ways: i) the receiving MN is first queried to establish it does not have the plugin already and ii) the plugin is flooded only to MNs that share the same platform with the owner of the plugin (this is necessary for heterogeneous environments with multiple platform configurations, such as our experimental testbed).

4 Usage scenario and testbed evaluation

For purposes of validation and experimentation we have implemented the proposed programmable middleware platform and deployed it in our experimental testbed. After reviewing the specific implementation details, we present the results obtained when testing our implementation in the testbed.

4.1 Testbed Configuration and Platform Implementation

To test the platform's performance and efficiency and also examine its operation in a real environment, we deployed it in our experimental MANET testbed that comprises 2 laptops and 4 PDAs (see Table 1 for configuration details). The testbed is a 6-hop MANET and is considered as a relatively reliable environment so that the results can be extrapolated and general conclusions can be drawn.

The platform is implemented using the Java 2 Micro Edition (J2ME). This version requires a much smaller memory footprint than the standard or enterprise edition, while at

the same time it is optimized for the processing power and I/O capabilities of small mobile devices. We also used the Connected Device Configuration (CDC) framework instead of the limited one (CLDC), as the latter lacks support for required advanced operations. We chose to use Java because of its ubiquity and platform independence. Our platform caters also for both Java and C/C++-based plugins. The use of Java requires MNs to have the Java Runtime Environment (JRE) installed. Although this is relatively memory-hungry, our hands-on experience confirms that even the resource-poor PDAs can comfortably support the execution of the JRE.

Table 1 Testbed hardware configuration

Platform	Configuration Attribute	Description
PDA	Processor	400 MHz Intel XScale
	Memory	48 MB ROM, 128 MB RAM
	Operating System	Familiar Linux 2.4.19
	Wireless interfaces	Integrated wireless LAN 802.11b
Laptop	Processor	1,7 GHz Intel Centrino
	Memory	512 MB RAM
	Operating System	Debian Linux 2.6.3
	Wireless interfaces	Integrated wireless LAN 802.11 a/b/g

The communication between MNs uses the lightweight XML-RPC protocol [17]. XML-RPC is a subset of the Simple Object Access Protocol (SOAP) with only basic functionality enabled. It allows software running on different operating systems and hardware architectures to communicate through remote procedure calls (RPCs). XML-RPC uses the HTTP protocol as transport and XML encodings for the RPC protocol itself. We chose an XML-based approach because we also use XML to represent contextual data collected by MNs. We could have possibly chosen Web Services, but this approach would have certainly been more heavyweight. In addition, Web Services, in the same fashion with distributed object technologies such as CORBA, necessitate object advertisement and discovery functionality, which is not required in our platform that relies on simple message passing modelled through RPCs. Given our recent performance evaluation of XML-RPC and other management approaches [16], we believe that XML-RPC provides a useful blend of functionality and performance.

Trivial FTP (TFTP) [18] was used for the distribution of the plugins. It is less complex than FTP and consumes less network resources. TFTP has no password-based user authentication, which saves both time and traffic in a trusted environment; as already mentioned, security in an ad hoc environment is an important issue but is outside the scope of the current work. In addition, TFTP uses only one connection, contrary to FTP that requires two connections, one for control and one for data traffic.

4.2 Autonomic routing protocol selection

The scenario we chose to test on our experimental testbed includes the dynamic change of the routing protocol used in the MANET. MANET routing protocol performance is

dependent on the stability of the network itself. Reactive routing protocols are better suited for very volatile network topologies, while proactive approaches for more static MANETs. The scenario implemented was that of the dynamic routing protocol change according to contextual information regarding the mobility of MNs. MNs use initially the reactive AODV routing protocol [19] for their ad hoc communication over 802.11b, while at some point indicated by the change in the mobility pattern they switch to the proactive OLSR protocol [20] as the network becomes close to stationary. This decision is derived and imposed by the MB. Both these routing protocols are realized as C-based user space daemons. Practical problems during this experiments included wireless link interference given that the wireless interfaces were in a confined space. In addition, since testing for various network topologies was necessary, we used a MAC address filter tool to emulate broken links or unreachable destinations.

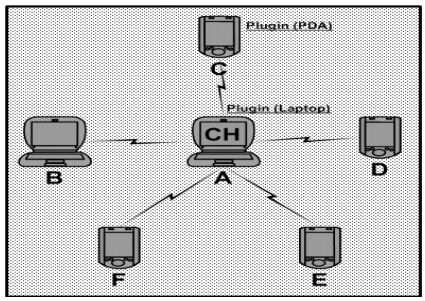
The scenario serves the purpose of presenting both the self-configuration and self-optimizing aspects of the platform, as well as the platform functionality. The self-configuration aspect is apparent from the scenario itself, while in this case the self-optimizing aspects refer to the fact that by changing the network protocol we achieve better performance of the MANET by means of bandwidth consumption (proactive and reactive routing protocols consume different amount of bandwidth and work better in different network states).

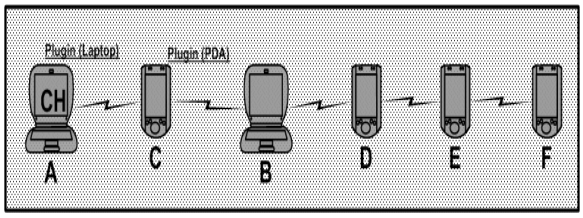
We experimented with many different topologies, routing protocols and other plugins to get a concrete understanding of the platform's operation. In the following subsections we present experimental results regarding the routing protocol switch scenario for three different yet representative network topologies: *star*, *random* and *line*. The star topology models a centralized approach, with the MB conveniently located in the centre and comprised of one node, having a 1-hop distance from other nodes. The line topology is the one that performs worse than the others, and models a sparse MANET with 6-hop diameter (the MB in this case is comprised of 4 nodes in a total of 6). The random topology models a middle-ground situation between the previous topologies and models the most common case real-world scenario (2 nodes form the MB). Although we have implemented context processing and dissemination in our platform, getting mobility information requires sensors MNs such as accelerometers, GPS support, etc. Given the practical difficulty of sensing real mobility changes, we chose to generate them artificially, through pre-specified timers and mockup context information. As we were mostly interested to assess the performance in terms of the plugin dissemination and activation, this approach is adequate. We plan though to focus on context-based performance issues in future work. Finally, it is essential to emphasize that the results have derived by a number of identical experiments and mean values are presented. Table 2 presents the results regarding the three described topologies as far as incurred traffic is concerned and convergence time.

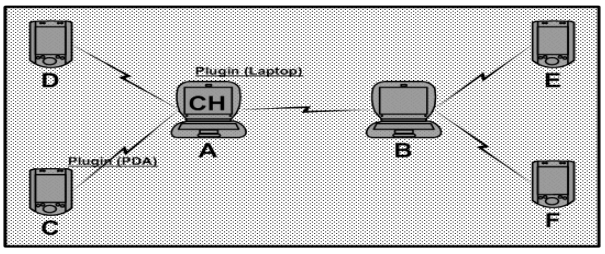
Results from testbed measurements prove first of all that the platform functions properly, since the routing protocol dynamic change performs smoothly and in accordance with the network mobility, while the situation can revert to the original configuration if the necessary conditions are met. The platform as evaluated in our testbed seems to fulfil its

goal as being lightweight and deployable on devices with limited resources, such as PDAs. The time needed for the initialization of the base functionality is 26 msec for the laptops and 741 msec for the PDAs, while the memory utilization was 3788 bytes and 4208 bytes respectively. The differences in time are attributed to the significantly different processing capabilities, while memory consumption is almost identical, which was expected since the platform is the same for both configurations.

Table 2 Experimental testbed results under various MANET topologies







Star Topology Time required for convergence: 41.96 sec
Routing related traffic: 7736 bytes
Inter-MN traffic: 41742 bytes
TFTP traffic: 1064880 bytes
The MB is formed of 1 node, solely A
Line Topology Time required for convergence: 47.94 sec
Routing related traffic: 14332 bytes
Inter-MN traffic: 83145 bytes
TFTP traffic: 1530924 bytes
The MB is formed of 4 nodes, C, B, D, E
Random Topology Time required for convergence: 44.43 sec
Routing related traffic: 12068 bytes
Inter-MN traffic: 51491 bytes
TFTP traffic: 1366896 bytes
The MB is formed of 2 nodes, A and B

The other parameters of the testbed experimentation prove the efficiency of the platform. From the moment the management body identifies the need to alter the routing protocol, up until the activation of the new routing protocol the time required is at acceptable levels, being dependent on the size of the routing plugin and the network size. The OLSR routing plugin has a size of 450 KB for the laptops and 98,1 KB for the PDAs. The convergence time required for the alignment of nodes capabilities depends on the distributed plugin. In

our test case the plugin size is significant, and thus requires considerable time for its deployment throughout the network. The measured time takes into account the fact the wireless links are not stable throughout the experiment due to interference reasons. In a number of experiments, link breakages occurred without any external intervention, and we attribute these to the inter-MN interference. Given these link breakages, the time measured in our experiments includes the additional latency introduced for route reconstruction.

Another important observation is the fact that the inter-MN traffic is rather limited with a maximum of 83145 bytes for the line topology, which is attributed to the fact that this is the sparsest one and the MB is composed of many nodes due to the specific node location. Even so, the inter-MN traffic is not large enough to make our hybrid management approach inapplicable. The inter-MN traffic includes the traffic required to construct and maintain the MB, the aggregated context advertisements from the MNs to the MB and other platform specific MN calls. Regarding the TFTP traffic this includes the transfer of the routing protocol plugin to the MNs that do not have it. This noteworthy traffic size is justified if one considers the significant size of the plugin and the fact that two versions are disseminated in the MANET (laptop and PDA versions).

5 Conclusions

We presented the foundations and major design principles of a context-aware, programmable middleware platform that enables self-configuration in MANETs. The platform has been implemented and successfully deployed on our experimental testbed, with encouraging initial results. Our future work focuses on further expanding the architecture to take into account more elaborate management policies that conform and adapt to the dynamic nature of the MANETs. We have limited our experimental evaluation of the platform to include only results from actual deployment on our testbed. We plan though to test its performance, scalability and its effect on MANET optimization using also simulation tools, complementing those MANET simulations with real-world practical experiments as suggested in [15]. Understanding the major security implications that may arise from the deployment of software modules on mobile nodes, we plan to expand our framework to incorporate advanced security mechanisms using possibly "sandbox" techniques for controlled execution in a failsafe environment and authenticated remote activation of software modules.

6 References

[1] Haas, R., Droz, P. and Stiller, B., "Autonomic service deployment in networks", *IBM Systems Journal*, Vol. 42, No 1, 2003

[2] Kephart, J.O. and Chess, D.M., "The Vision of Autonomic Computing", *IEEE Computer*, January 2003

[3] Ganek, A. G. and Corbi, T.A., "The dawning of the autonomic computing era", *IBM Systems Journal*, Vol. 42, No 1, 2003

[4] Crawford, C.H. and Dan, A., "eModel: Addressing the Need or a Flexible Modeling Framework in Autonomic Computing", *10th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS02)*, October 2002
[5] Dong, X., Hariri, S., Xue, L., Chen, H., Zhang, M., Pavuluri, S. and Rao, S., "AUTONOMIA: An Autonomic Computing Environment", *IEEE International Conference on Performance, Computing and Communications*, April 2003
[6] Chadha, R., Cheng, H., Cheng, Y.-H., Chiang, J., Ghetie, A., Levin, G., and Tanna, H., "Policy-based mobile as hoc network management", *5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY04)*, 2004
[7] C. Tschudin, H. Lundgren, H. Gulbrandsen, "Active Routing for Ad hoc Networks", *IEEE Communications*, Vol. 38, No. 4, April 2000.
[8] C. Bohoris, A. Liotta, G. Pavlou, "Evaluation of Constrained Mobility for Programmability in Network Management", *11th IEEE/IFIP Int. Workshop on Distributed Systems: Operations and Management (DSOM'00)*, December 2000.
[9] G. Goldszmidt, Y. Yemini, "Evaluating Management Decisions via Delegation", *Proc. of IEEE Integrated Network Management III*, pp. 247-257, Elsevier, 1993.
[10] A. Malatras, G. Pavlou S. Gouveris, S. Sivavakeesar and V. Karakoidas, "Self Configuring and Optimizing Mobile Ad Hoc Networks", *to appear as a short paper in the Proceedings of the IEEE International Conference on Autonomic Computing (ICAC 2005)*, June 2005
[11] S. Gouveris, S. Sivavakeesar, G. Pavlou and A. Malatras, "Programmable Middleware for the Dynamic Deployment of Services and Protocols in Ad Hoc Networks", *to appear in the Proceedings of the IEEE/IFIP Integrated Management Symposium (IM 2005)*, May 2005
[12] P.-J. Wan, K. M. Alzoubi and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks", *IEEE Infocom 2002*
[13] R. Friedman, M. Gradinariu and G. Simon, "Locating cache proxies in MANETs", ACM MobiHoc 2004
[14] U. Kozat and L. Tassiulas, "Network layer support for service discovery in mobile ad hoc networks", *IEEE Infocom 2003*
[15] Tschudin, C., Gunningber, P., Lundgren, H., Nordstrom, E., "Lessons from experimental MANET research", *Ad Hoc Networks*, Vol. 3, Issue 2, pp.221-233, March 2005, Elsevier, 2005
[16] G. Pavlou, P. Flegkas, S. Gouveris, A. Liotta, *On Management Technologies and the Potential of Web Services*, IEEE Communications, special issue on XML-based Management of Networks and Services, Vol. 42, No. 7, pp. 58-66, IEEE, July 2004.
[17] XML-RPC specifications web site, http://www.xmlrpc.com/spec, accessed April 2005
[18] K. Sollins, The TFTP Protocol, IETF RFC 1350, July 1992
[19] C. E. Perkins, E. M. Belding-Royer, and S.R. Das, *Ad hoc On-Demand Distance Vector (AODV) Routing*, draft-ietf-manet-aodv-13.txt.
[20] Clausen, T., Jacquet, P., *Optimized Link State Routing Protocol (OLSR)*, RFC 3626, October 2003.