A Programmable Framework for the Deployment of Services and Protocols in Mobile Ad Hoc Networks

Apostolos Malatras, George Pavlou, and Sivapathalingham Sivavakeesar

Abstract—Mobile ad hoc networks (MANETs) are characterized by their heterogeneity and the diverse capabilities of their nodes given that almost any device with a wireless network interface can join such a network. In such an environment it is difficult to dynamically deploy services and protocols without a common understanding among the participating nodes and their capabilities. A deployment/provisioning framework must cope with the high-level of device heterogeneity, degree of mobility, and should also take into account the potentially limited device resources. This paper presents a context-based programmable framework for dynamic service/protocol deployment that allows the nodes of a mobile ad hoc network to download and safely activate required service/protocol software dynamically. Downloading and activation can be triggered through preconditions evaluated according to available contextual information. This strategy leads to the alignment of the nodes' capabilities so that common services and protocols can be deployed even if they are not available at every node. In addition, dynamic context-driven deployment may lead to a degree of network self-optimization. We present the programmable framework and functionality and evaluate its various aspects through testbed experimentation, simulation and analytical modeling. The results demonstrate good performance with respect to the supported functionality.

Index Terms—Programmability; System framework; Ad-hoc network and service management; Autonomic operation.

I. Introduction

The concept of mobile ad hoc networks (MANETs) has recently received significant attention due to the increasing popularity of tetherless computing and the rapid growth of wireless networking. In ad hoc networks, mobile nodes are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. An ad hoc network typically operates in a standalone fashion but may also be connected to an infrastructure-based network through a gateway, e.g. a WLAN access point, a cellular network base station, etc. Conventional wireless networks require as prerequisite some form of fixed network infrastructure and centralized administration for their operation. In contrast, since MANETs are self-creating, self-organizing and self-administrating, mobile nodes are responsible for dynamically discovering other nodes they can communicate with. This dynamic network formation that encompasses heterogeneous devices greatly benefits from the ability to rapidly create, deploy and manage services and protocols in response to user demands.

The dynamic deployment of services and protocols in heterogeneous MANETs has not been adequately addressed in the literature. This is partly due to the fact that MANET applications have so far been envisioned mostly for emergency and military situations with pre-existing functionality. However, future MANETs may support a variety of applications in different environments and should allow for the dynamic tailoring of capabilities according to arising needs, possibly in a dynamic fashion based on contextual information. For example there are many potential solutions for ad hoc networks regarding routing protocols, quality of service (QoS) enhancements, etc., that typically depend on the characteristics of the particular ad hoc network, e.g. topology volatility, characteristics of radio links, node capabilities, etc. Given the multitude of potential solutions, programmability is of paramount importance to allow mobile nodes to be enhanced on the fly with the required capabilities in the ad hoc environment. In addition, application servers may migrate to more powerful devices that have the required processing capacity while less powerful devices may outsource computing tasks. Programmability is possible through recent advances in distributed systems technologies and transportable "execute-anywhere" software. In this paper we present a novel programmable platform approach that can support cooperation, adaptability and alignment with respect to the required basic capabilities and additional services of the participating nodes in a secure manner. This cooperation and switching to a different mode of operation may be triggered dynamically, allowing a degree of self-management to be achieved.

A key aspect of MANETs is the possibility to adapt to their environment through context awareness for many cooperation and coordination scenarios. Context-information may be used to trigger network changes according to predefined rules, leading to dynamic and, to an extent, autonomic decision-making. The sheer amount of context information necessary for relevant adaptation places an important burden on the network, given that potentially large amounts of data from diverse sources need be managed. This requires an infrastructure for sensing, collecting, summarizing and making context information available [12]. The proposed framework incorporates such functionality [22] but in this paper we concentrate mostly on the programmable platform aspects for node capability alignment, including an exemplary case-study of dynamic switching between a proactive and a reactive routing protocol. The work presented here enhances and extends our initial work on programmability in ad hoc networks presented in [18]. The work in this paper includes extended

Manuscript received February 5, 2007; revised February 11, 2007. The associate editor coordinating the review of this papr and approving it for publication was J. W-K. Hong.

The authors are at the Centre for COmmunication Systems Research, Department of Electronic Engineering, University of Surrey, U.K. (e-mail: {A.Malatras, G.Pavlou, S.Sivavakeesar}@surrey.ac.uk).

Digital Object Identifier: 10.1109/tnsm.2007.021108

testbed experimentation with mobile nodes of diverse hardware/software platforms and capabilities and also analytical modeling of the proposed plugin distribution scheme. Extended simulations have also been performed.

The rest of the paper is organized as follows. We present related work on programmability and dynamic service deployment. The proposed platform functionality and architecture are described, also discussing relevant design issues. We present and evaluate an application case study for the proposed platform while we then evaluate the proposed approach, first through testbed experimentation and then through analytical modeling and simulation. Finally, we conclude the paper and present directions for future work.

II. Related Work

The dynamic nature of MANETs and the relevant variable conditions raise the need for runtime reconfiguration and node capability alignment. Programmability is a means to achieve this alignment and is essential in a MANET given the multitude of potential solutions for routing, quality of service and other protocol or application services. There exist various approaches to achieve programmability. The Active Networks approach advocates the dynamic deployment of new services by introducing "active packets" that carry logic as well as data. Code carried by active control packets is evaluated in routers [4] and this approach has also been used for active routing in MANETs [5]. The Smart Packets project [14] used active packets to improve the performance of large networks by delegating management decisions closer to the managed nodes.

Programmability is also possible through the provision of suitable management interfaces for modeling communication hardware/software in switches and routers; such interfaces allow code to be uploaded to network nodes and activated in a controllable fashion. This approach was first adopted in the Xbind framework, targeting the quick and flexible introduction of new telecommunication services in programmable network infrastructures [8]. Xbind has given rise to the IEEE P1520 initiative for Programmable Network Interfaces [9]. The Mobiware approach relied on Xbind, modifying and extending it for programmable cellular networks [10]. Mobile agents may be also used in full mobility scenarios, carrying code and state to manipulate different network nodes, or in a constrained mobility mode [6], which can be seen as a flexible approach for "management by delegation" [7]; in the latter case, code is uploaded and executed in network nodes through "elastic management agents," augmenting the node's functionality.

Relevant work related to MANETs includes middleware-based solutions for network programmability and dynamic service provisioning. The SensorWare framework [15] is middleware based on active networking principles that supports the programmable management of sensor nodes through lightweight, mobile control scripts. [19] defines a service efficiency parameter to characterize dynamic service deployment in MANETs; this work addresses the important issue of dynamic service deployment/termination in mobile nodes so that better service coverage is obtained at a minimal control cost.

It should be mentioned that while there exists research work on network programmability, there has been no previous attempt to apply it to ad hoc networks in the manner proposed here, i.e. not limited to a specific application domain. We should also mention that there exists work on hybrid and adaptive routing strategies similar to the one we propose in our experimental case study [23, 24]. The proposed hybrid routing approaches have not been implemented and deployed in real-world MANET environments, so there are still open questions regarding their applicability. We distinguish ourselves from this work by providing a generic framework that can cater for any protocol/service deployment in a MANET and not just routing protocol switching. We do not claim to have provided a novel routing scheme for MANETs; the specific case study simply motivates the framework's operation. The latter is not restricted to dynamic routing strategies as it is the case in related work limited to a specific application domain, but allows for dynamic strategies of diverse nature to be deployed in MANETs in a generic fashion.

III. Programmable Ad Hoc Framework: Functionality and Architecture

The programmable platform we propose uses a lightweight approach through loadable plugins that can be installed in nodes to extend or alter their capabilities. Examples of such plugins include new routing protocols, extensions to existing ones or any other service/protocol that might be required. Plugins essentially form the system's functionality. A plugin is a piece of Java or C++ code with well-defined interfaces as required by the platform infrastructure; in its simplest form it may even be a script with configuration commands to be used in delegated configuration scenarios. In order to decide on a particular plugin instance for a given scenario, a plugin election takes place utilizing current context information. This involves advertisements beforehand by every Mobile Node (MN) about the plugins it owns. The elected plugin is then distributed throughout the MANET in a peer-to-peer fashion, using flooding initiated from the nodes that have it. Following its installation, it is activated in all the nodes. The rest of this section discusses design issues of the proposed framework and presents initial reasoning behind some of the design choices. Security is of major importance to MANETs, especially given the nature of our programmable platform approach. We present briefly relevant mitigation techniques but a detailed analysis and evaluation lies outside the scope of this paper.

The proposed framework can support any type of service/protocol deployment, assuming services or protocols are realized as plugins executed in user space. However, as detailed in Section IV that describes the application scenario, the work presented aims at network-level services, dynamically deploying and switching MANET routing protocols on the fly.

A. Hierarchical vs. Distributed Management

A key consideration in MANETs is the organizational model to be deployed. In order to cope with potentially large scale, a common practice is to organize the MANET into clusters, each managed by an elected local leader or Cluster Head (CH). Assuming a hierarchical approach, CHs then cooperate and either elect a global leader or Network Head (NH) [1] or a set of CHs that collectively undertake the role of NH. The NH or the set of CHs that undertake the NH role take key management decisions, such as triggering and coordinating plugin distribution and activation. The election process for the CHs and the NH guarantees reliability by also electing backup nodes that can assume the roles of these entities in case a node departs the network or fails for unforeseen reasons. The CH election process is re-instantiated upon the occurrence of significant topological changes.

This hybrid hierarchical approach that we have adopted for our design (hybrid in the case when many CHs cooperate to take collective decisions) is similar to that of routing protocols, e.g. OSPF, and scales well, limiting interactions within a cluster or among cluster heads. It also allows operation in a controlled distributed fashion, when decisions are not taken by a single NH, but through cooperation and "voting" among the CHs. A diametrically different approach is a fully dis-

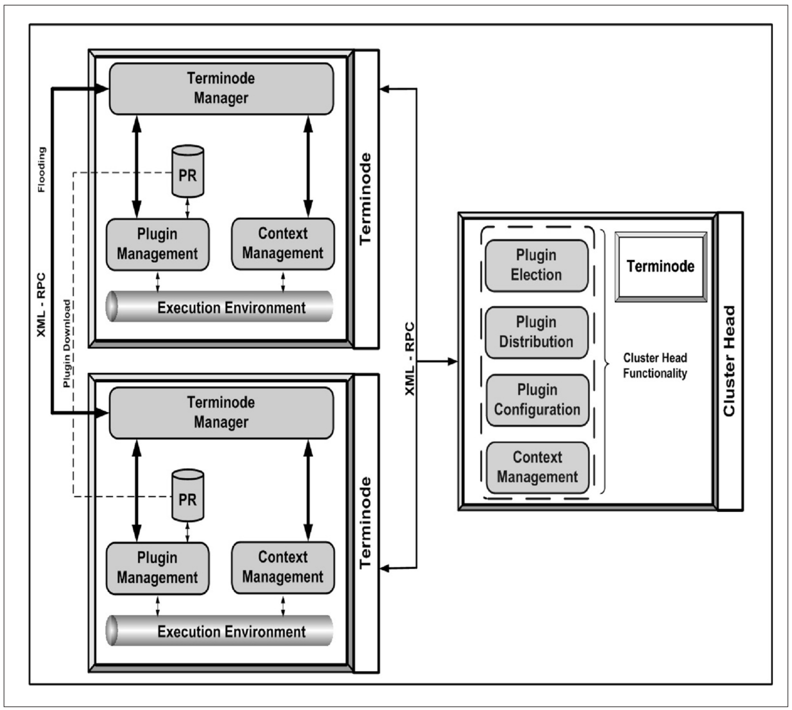

FIGURE 1. *Interactions between platform components within a cluster.*

tributed one, in which all the mobile nodes have “equal rights” and determine collectively any management decisions. This approach requires more complex cooperation protocols and does not scale for large networks.

The CHs/NH election process is based on contextual information such as location, capabilities (processing power, memory, battery life, node mobility etc.), expected residence time in the current location/cluster and user privileges. These are collected from sensors embedded in the mobile nodes. For example, the CH/NH needs to be a relatively central node in the cluster or network respectively while there is no point in electing a CH with high probability to move away from its current location soon. In addition, the candidate CH nodes should be “thick” in terms of capabilities. There exists a lot of work in the literature on cluster formation and CH/NH election and we have also proposed relevant algorithms. Stable cluster formation is important, hence the CH election heuristic can be similar to the one we proposed in [1] for longer-term large-scale MANETs or similar to the one we proposed in [2] for more spontaneous MANETs. In [1] and [2] we compared our CH election heuristic to other similar approaches, and showed that our heuristics result in more stable clusters. In short, the CH/NH election procedure is based on building a stable connected dominating set of the MANET underlying graph. For details on the reliability and resilience of this process and relevant performance analysis we refer the reader to [25].

B. Platform Communication and Components

We identify two fundamental entities in our programmable platform architecture, the CH and the Terminode modules depicted in Fig. 1. The NH acts as a head for the set of cluster heads in the MANET, so its role is similar to that of a higher-level CH. We have currently only implemented the hierarchical model among CHs in which the NH takes all management decisions but we also plan to investigate a collaborative peer-to-peer model among CHs in the future. Depending on the status of a mobile node, its functionality can switch among Terminode at the lowest level of the hierarchy, CH a level above and NH as a top-level cluster head.

The CH module is responsible for the management of the programmable Terminodes regarding plugin election and distribution, as well as (re-) configuration. The key components of the CH module are:

- Plugin Election: it is responsible for the initiation and coordination of the election process. It receives the plugin advertisements from the nodes containing the characteristics of the plugins they hold, elects one and informs the nodes about the elected plugin. A necessary requirement for a plugin to be elected is that versions for every available node platform exist.
- Plugin Distribution: this component manages the distribution of the elected plugin to the nodes. The distribution is performed using a controlled flooding scheme, in which a node is first queried to examine if it already has the plugin and if it can support its execution, given the heterogeneity of mobile nodes in terms of hardware/software. This component takes also care of plugin activation.
- Plugin Configuration: this component handles the (re-) configuration of plugins that have already been deployed on the MANET.
- Context Management: this component manages context information gathered from nodes. The latter send aggregated context information to the CH, so that the CH/NH can identify the need for configuration changes, for example the relative mobility of nodes within a cluster in our working case study presented in section IV.

The Terminode module deals with the collection /aggregation of local context information at a node (e.g. tracking node mobility) and handles the management of the loadable plugins locally through advertisement, storage, distribution, installation and configuration. The specific components of the Terminode module include:

- Plugin Repository (PR): this stores actual plugins and their properties. The plugin properties are advertised to the CH/NH and include the plugin execution platform and functional requirements for its operation.
- Plugin Management: it provides all the necessary operations to access the repository and retrieve/update plugins and their properties. It is responsible for advertising the available plugins to the CH/NH Plugin Election component when asked and waits for requests from the CH to distribute a plugin to its neighbors. Plugin installation, execution, (re-)configuration and termination are also amongst its responsibilities.
- Context Management: it handles the sensing, collection, modeling of context information locally at the MN and preprocesses/aggregates this information in order to send it to the respective CH. Aggregation results in reduced traffic and context processing at the CH. Monitoring aspects of context gathering and dissemination are not considered in this paper since the emphasis is on the dynamic, programmable framework, but we have presented relevant work in [22].

The platform components communicate using XML-RPC. We discuss the choice of XML-RPC along with other implementation decisions later.

C. System Operation

We describe here the complete system operation, from election triggering to plugin activation, assuming a network of one cluster for simplicity. The CH is responsible for the plugin election process. Based on the aggregated context from Terminodes and predefined rules, the CH decides when to trigger the election process. It does this through the Plugin Election component and decides on the type of the plugin currently required by the MANET, e.g. the requirement to switch to a reactive routing protocol.

The Plugin Election component contacts the cluster Terminodes for advertisements of candidate plugins. Each node performs a lookup in its repository for suitable plugins that satisfy the CH requirements, e.g. reactive routing protocols, and replies to the CH with the characteristics of the retrieved plugins. The CH Plugin Election component executes the election process and decides on the most suitable plugin, based on the current context. It should be noted here that the set of candidate plugins is pruned so as to include only those that are either executable in all available node platforms (e.g., Java-based plugins), or those that are platform specific (e.g. C/C++-based) but an implementation for each configuration exists in the MANET. The main reason for this is to avert a situation where the selected plugin cannot run on every available node platform and thus node alignment is compromised.

Following the actual plugin election, the CH contacts Terminodes that already own the elected plugin and instructs them to distribute it to their neighbors. Plugin distribution is carried out in a peer-to-peer flooding fashion, with the "owning" nodes passing the plugin to their adjacent nodes, and so on. Prior to any plugin exchange, each node probes its peers to find out if they have already acquired the plugin and subsequently avoid unnecessary transmissions. This probing also guarantees that the correct plugin version is distributed — this aspect is important for platform-specific plugins. For the actual transmission, a Terminode contacts its adjacent Terminode (through their Plugin Management components) and passes all the characteristics of the elected plugin, followed by the plugin itself.

When a new plugin has been successfully installed, the Terminode Plugin Management component sends a relevant notification to the CH. At this point, the plugin is installed and available to be activated. The CH Plugin Election component, after receiving installation notifications from all Terminodes, disseminates an activation message across the cluster to instruct the member nodes to start executing the elected plugin. Each Terminode then picks up the plugin from its repository and executes it in user space. This ensures "weak" synchronization of plugin activation amongst all Terminodes and also node alignment, since activation occurs only when confirmation of successful installation of the plugin by all Terminodes has been received. In case of error, the process is restarted e.g. to take into account significant topological changes that might have taken place. Failsafe mechanisms presented in section III-F, have been considered in order to ensure correct operation.

D. Loadable Plugins

Plugins can be loaded, activated, re-configured and removed dynamically. The plugins are essentially loadable objects or modules with well-defined interfaces that allow their controlled execution in the surrounding platform environment. A plugin can be any type of platform-independent executable object, e.g. Java-based, or platform-dependent, e.g. C/C++-based. Our platform supports both but platform-dependent C/C++ plugins can only be executed on the hardware/software platform combination for which they were compiled. It is obvious that the Java plugins are more generic, extensible and platform independent; on the other hand, platform-dependent C/C++ plugins are important for services and protocols for which performance is crucial. Programmable plugins are loaded and activated in user space as general purpose processes. Plugins as operating system kernel extensions are both difficult to engineer and also exhibit inherent security/instability problems. Plugins expose two programmable interfaces for the purpose of configuration and monitoring. The first interface is used to configure and alter aspects of the plugin functionality at run-time. On the other hand, through the monitoring interface plugins can provide information regarding their status and various useful statistics on their operation. Due to the dynamic nature of plugin installation and activation, security is of paramount importance to ensure uncompromised operation and MANET stability. We briefly examine security implications and mitigation techniques later.

E. Plugin Election

The NH is responsible for the election process and the identification of the need for configuration changes. From the collected context information and predefined policy rules, it realizes what type of configuration change is needed. Details on the context-driven policy-based MANET management architecture are outside the scope of this paper (we refer the reader to [25]), which focuses on the programmable platform aspects. It should be also noted that relevant input triggering a configuration change is also possible by a human manager interacting with the NH node, e.g. for emergency and military scenarios. Having decided on the necessary configuration change, the NH queries the nodes regarding available plugins of the specified type. The nodes respond accordingly and the NH builds a list of the available plugins of that type that exist in the MANET. It then elects one plugin among the candidate ones and asks for it to be deployed throughout the network for the required changes to take effect.

The election process utilizes Equation 1 and considers plugin properties. Based on the latter, the candidate plugin set is pruned so as to only include plugins that can operate in all nodes as explained. The election process uses selection criteria associated with plugin technical details. These criteria are assigned a unique identifier and have comparable values. Examples of such criteria we consider in our design include the CPU cycles required, the run-time memory and the plugin storage size. Each criterion is assigned a weight based on its importance in the election and its values are "normalized" in a common-ground range for comparability between different plugin/platform combinations. The weights are adaptive and can be varied based on context information, e.g. the scarcity of resources. The values of the weights are uniformly initialized for all nodes of the MANET and only the NH is responsible for their alteration.

$$f(x) = \sum_{i=1}^{n} w_i \times A_i(x), \tag{1}$$

where: $i \in [1, n]$ refers to the election criterion, A_i is the criterion's value i for plugin x (range normalized to [1, 10]), $w_i \in [0, 1]$, $\Sigma w_i = 1$ and $A_i \in [0, 10]$, $\forall i \in [1, n]$.

Let's consider an example. Based on context information, the decision to deploy a reactive routing protocol is reached. Assume there exist three implementations of such protocols in the MANET: a cross-platform implementation of AODV and DSR and a platform-specific implementation of TORA; these three form the set of possible plugins. The TORA plugin is "pruned" because it is platform-specific. The choice between AODV and DSR depends on the value of $f(x)$ in equation 1. The criteria used are the storage size (A_1) and required run-time memory (A_2), with weights 0.3 and 0.7 respectively. The plugin for which f yields the smallest value will be used, given that it will be conceptually the most lightweight. The existence of relevant criteria in f and the relevant weights can also be adaptive as already mentioned.

F. Security and Reliability

The necessity to cater for security and reliability in our framework is evident. We can identify a number of potential problems that may arise. While a full scale mitigation strategy of relevant issues could form the core of a standalone paper, we briefly present the considerations we have taken into account to tackle security issues. We do not claim that the proposed measures constitute a complete solution for the various problems. They provide though our platform with sufficient robustness and ensure correct functionality under unstable conditions.

The distribution of plugins, i.e. executable code, among the nodes of the ad hoc network is a major security concern. An attacker could exploit this functionality and introduce malicious plugins, compromising the correct operation of nodes. This threat can be generalized to the assignment of the CH and NH roles. If an attacker were to compromise the CH/NH or masquerade itself as one of those then the operation of the network could be thwarted. We manage these security implications with a strategy based on establishing trust among network nodes, as proposed in [27]. When such relationships have been established then issues such as malicious nodes or fake CHs/NH can be avoided. Trust relationships can be established during the clustering process as suggested in [28]. In parallel, we employ a distributed security model on the ad hoc network, as proposed in [26]. This is based on principles of threshold cryptography and in short refers to a distributed, resilient public key scheme. Plugin distribution is then performed, only when proper authentication of the plugin owner has been established. The plugin itself is being digitally signed by its owner, hindering thus compromise of integrity and preventing malicious code to be activated on nodes. The only code that is installed has originated from trusted nodes. We have not yet implemented these security measures in our platform but this is a priority for our future research and development work.

The main concern regarding reliability in our design refers to the instability of the ad hoc network itself. Node movement can lead to cases where capability alignment cannot occur, because some nodes have moved in or out of communication range and thus the conditions that triggered the need for configuration changes have changed. Our platform is dynamic and adaptive, monitoring node context and cluster topological information constantly. This implies that changes in the ad hoc network formation are identified and thus the membership and context information is always up-to-date. In the case of node movements affecting the functionality of the platform while capability alignment is being employed, the reliability of the system is guaranteed with the use of failsafe mechanisms. By this we mean that an operation is not completed until acknowledgment by the majority of the participating nodes is received by the CH/NH — the majority threshold is adaptable and depends on the overall network mobility ratio. If for any reason the deployment of a new plugin fails then the MANET nodes roll-back to the last known working configuration. In the case of nodes departing, the CH/NH keeps track of the nodes that did not align themselves to the capabilities of the majority (through the clustering process) and if they rejoin the ad hoc network, in the next monitoring cycle they are explicitly instructed to configure themselves accordingly. If new nodes appear in-between configuration changes, then their capabilities will be aligned accordingly in the next monitoring cycle.

IV. Application Case-Study

Although the proposed platform can support service and protocol deployment at any level, we present here an example application scenario that addresses network-level protocol deployment. The use of a complicated scenario like this is by itself a novelty of our work, since to our knowledge there has been no similar experimentation effort in a real testbed. In this case, based on the current context, a routing decision is made to dynamically deploy and switch to a new MANET routing protocol on the fly. The NH decides based on context information whether the current routing protocol should be changed for more efficient operation of the whole network. Although the term context is broad in concept, here we use as context information the mobility of nodes in terms of their relative velocity. Based on this information, the NH checks whether the current routing protocol needs to be switched from reactive to proactive and vice-versa. Once the decision is made, the specific protocol — among the available routing protocol implementations — is selected through the plugin election process described earlier.

The primary goal of any ad hoc network routing protocol is correct and efficient route establishment between a pair of nodes so that packets are delivered in a timely manner. Route construction should be performed with a minimum of overhead in terms of bandwidth consumption. Many different protocols have been proposed to solve the MANET multihop routing problem, each based on different assumptions. There exist mainly two different categories of relevant protocols: proactive and reactive (or on on-demand). The proactive ad hoc routing approach is similar to some extent to routing schemes for fixed networks and does not take into account the frequency with which routes are required. It relies on an underlying routing table update mechanism that involves the constant propagation of routing information throughout the network so that routes are always available. This is not the case, however, in reactive routing protocols, in which a route to a new destination is only established when the need arises to communicate with it; this obviously introduces latency.

Hence, different routing protocols provide optimal performance under different operating conditions. In the case of a MANET in which the nodes are relatively stationary and the topology changes rather infrequently, the proactive routing approach is more cost-effective in terms of routing overhead, and leads to improved performance in terms of the latency involved to transmit packets and the packet delivery ratio. On the other hand, when the mobility patterns of nodes are unpredictable and the network topology is very fluid, the reactive approach may lead to better performance. Hence, it would be advantageous to change the routing strategy depending on the mobility of the nodes. In our system mobile nodes can switch between the OLSR (Optimized Link State Routing) [20] and the AODV (Ad Hoc on Demand Distance Vector) routing protocol [11]. OLSR is a proactive routing scheme while AODV is an on-demand routing one. When the NH determines that the relative velocity of the nodes of the network exceeds a certain threshold, it triggers protocol switching from a proactive to a reactive approach and vice versa. Every node monitors its own mobility with respect to its adjacent nodes and relevant summarized information is passed to the CH and subsequently to the NH that makes the decision. The exact aspects of context management, dissemination and the policy rules governing this switching are outside the scope of this paper, please refer to [22] for further details. The comparison we provide simply serves as a motivation for the viability of our approach and does not imply the introduction of a novel routing scheme.

Before proceeding in presenting a comprehensive evaluation analysis of our proposed generic framework for the deployment of services in MANETs, we deem necessary to assess the performance gain regarding the specific case study. We use GloMoSim simulations as the tool for our analysis and we attempt to evaluate the gain in network performance when using our proposed adaptive routing strategy in compar-

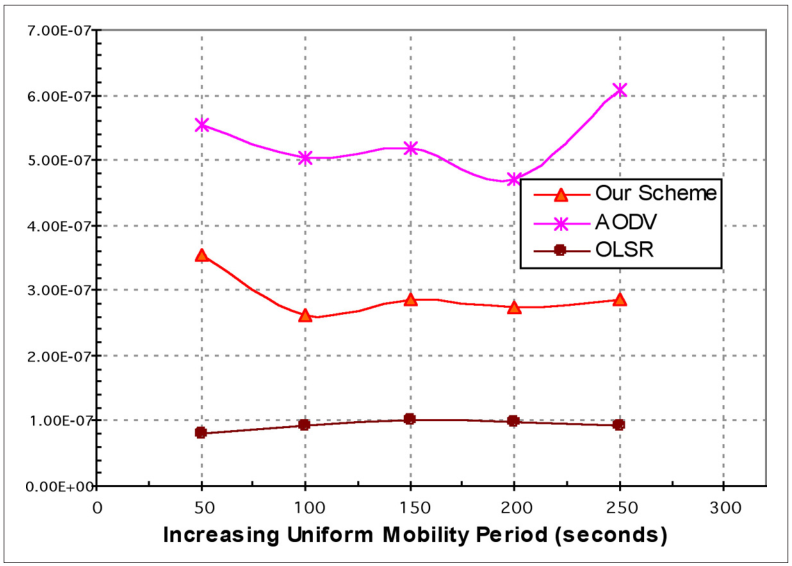

FIGURE 2. *Delivery ratio per control cost incurred vs. increasing UMP.*

ison to using a static one, in terms of control cost, packet delivery ratio for each data packet delivered and average end-to-end delay for each application-level packet. The packet delivery ratio is defined as the total number of data packets delivered to respective destinations divided by the total number of packets generated in the network.

We considered 100 nodes contained in 9 clusters, each with a dimension of 200×200 m^2 within the terrain area of 600×600 m^2. Traffic was generated using random CBR connections with a payload size of 512 bytes. At any point in time the total number of CBR source-destination pairs is kept constant (actually 40), and each session lasts for a time period that is uniformly distributed between 20 and 40 seconds. Each scenario was executed for 300 simulated seconds.

Since in MANETs communications often take place within smaller teams that tend to coordinate their movements, the group mobility model is a reasonable assumption in many application scenarios. If we assume that node mobility is completely unpredictable, it is impossible to address the issue of group division and the decision regarding protocol switching. For this purpose we developed a new group mobility model based on the classical random-waypoint model. In this model, we have two mobility patterns for the mobile nodes (i.e. uniform and random) and respectively two mobility periods. During the uniform mobility period (UMP), the relative velocity of any node with respect to any other node in the network takes a constant value. On the other hand, during the random mobility period (RMP), the relative velocity of each node with respect to each other is random. During this period (RMP), the random-waypoint mobility model governs the mobility pattern of each node, and hence this results in varying cluster membership of nodes. The RMP and UMP occur interchangeably at the same time for all the network nodes. The time period for both UMP and RMP is based on an exponential distribution with a mean value varied during each run of simulations as described later. The need for this new group mobility model was due to our desire to simulate context conditions that trigger protocol switching.

The NH will decide whether to trigger routing protocol switching based on the relative mobility patterns of nodes. When the relative velocity of the nodes with respect to their CHs tends to be almost constant (i.e. during UMP), the NH will make the decision to switch to a proactive routing approach, i.e. OLSR. On the other hand, when the relative velocity of the mobile nodes takes random values, the NH will attempt to switch to a reactive routing approach, i.e. AODV. We compared the performance improvement of our approach, where the routing protocol switches between AODV and OLSR in an adaptive, context-driven, dynamic fashion, with an approach where the routing protocol is static.

With Figure 2 we attempt to assess the effect of the mean value of the UMP on the normalized throughput (packet delivery ratio) per control cost under different traffic scenarios. In this process, the UMP mean is increased from 50 to 250 seconds, while the RMP mean takes a constant value of 100 seconds. As it was previously explained, the proactive routing approach involves constant propagation of routing table information, whereas this is not the case with the reactive routing one. As a result, the proactive approach incurs substantial amount of routing-related control cost in comparison to the reactive approach. As it can be seen from Figure 2, the packet delivery ratio per control cost incurred is lower for OLSR in comparison to AODV. As it is expected, the performance of our scheme lies in-between the performance of AODV and OLSR.

Figure 3 depicts the average end-to-end delay each application packet experiences as a function of increasing UMP under different traffic scenarios. This is, however, measured for successfully received packets only, and hence it does not reflect the actual delay; for instance, when all the packets between a given source-destination pair are dropped due to the inability of the routing protocol to find routes, the end-to-end delay appears to be zero. From Figure 3 it becomes clear that the delay improves with OLSR when the UMP increases. The delay of AODV appears to be poor when compared to

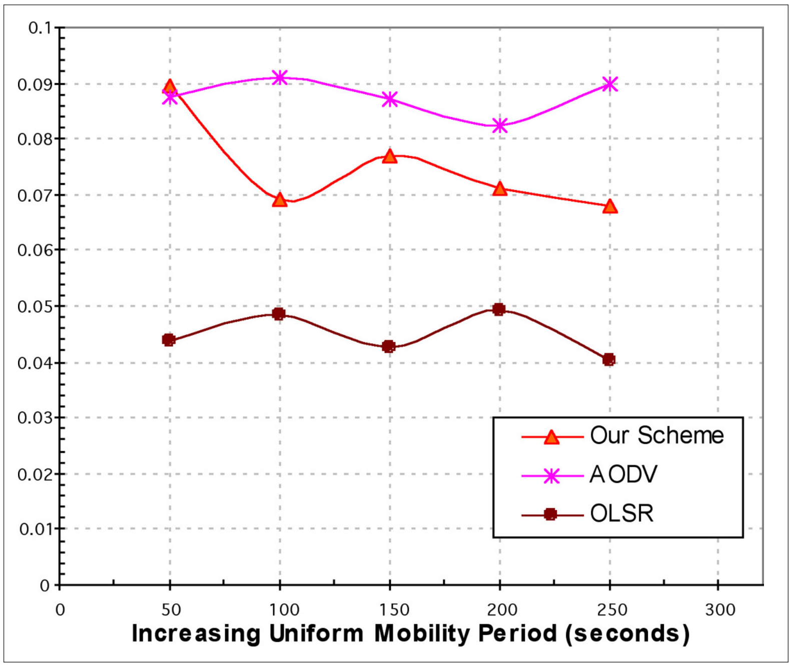

FIGURE 3. *Average end-to-end delay of a successfully received packet vs. increasing UMP.*

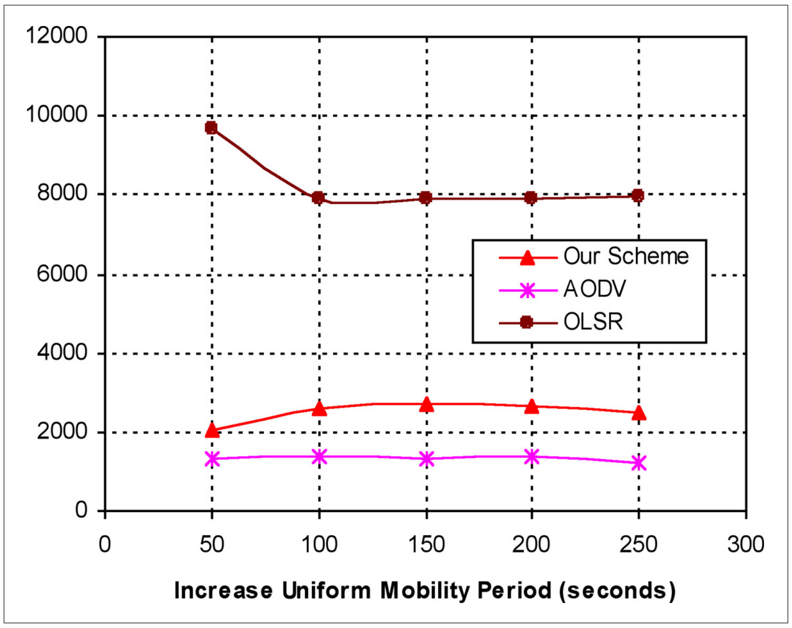

FIGURE 4. *Control cost incurred vs. increasing UMP.*

OLSR as the UMP increases. Since in our scheme, the routing protocol switching between AODV and OLSR depends on the mobility patterns, the delay lies in-between those of AODV and OLSR, as expected. The efficiency of OLSR in comparison to AODV was expected, since being a table-driven routing protocol means that active routes are maintained for every destination. In the reactive approach, the route has to be established on-demand, resulting in increased delay. Our scheme utilizes both reactive and proactive principles and thus its performance lies in-between that of AODV and OLSR.

Figure 4 depicts the overall control cost incurred as a result of the employed routing strategy. The table-driven OLSR yields significant control traffic, which is even higher when mobility increases, i.e. when UMP is small. On the other hand, the reactive AODV builds routes on-demand and thus the control overhead is relatively small. Our hybrid strategy outperforms OLSR and incurs control traffic almost comparable to that of AODV.

It is evident from the simulation analysis that no routing strategy is a panacea for all routing cases in MANETs, since different approaches have their own benefits and drawbacks. Our hybrid scheme achieves satisfactory results under varying conditions. While OLSR exhibits much less delay than AODV and the exact opposite is the case as far as control cost is concerned, our scheme is considered as a viable alternative and performs in between these two approaches in all cases. This is demonstrated in the graphs presented above. The obvious advantage and merit of our approach is that it is generic and can lead to optimized hybrid strategies for various application scenarios and not only for a particular application as in [23]. The routing protocol case study presented only serves as a proof-of-concept of the platform's operation and motivates its usefulness: it is not meant as a novel, more efficient routing strategy, as this would require a much more comprehensive study.

V. Platform Evaluation

For purposes of validation and experimentation we have implemented the proposed programmable platform and deployed it in our experimental testbed. After discussing the relevant implementation details, we present the results obtained when testing and validating our implementation in the testbed. This section concludes with simulation results and analytical modeling, examining the efficiency and scalability of our approach. It should be noted that the three different techniques we used to evaluate the platform consider the same set of metrics i.e. control traffic overhead and convergence time, each one thus validating the other. Testbed experimentation proves the viability of our platform in a real-world operational environment. On the other hand, scalability cannot be addressed through experimentation due to the fact that one cannot expect a testbed of hundreds of nodes. The need for simulation analysis is thus evident. Analytical modeling is used to mathematically model the performance of our platform and to validate the results obtained from testbed experimentation and also simulation. We therefore consider our approach comprehensive and holistic since it provides an overall evaluation of our platform through a variety of assessment techniques.

A. Testbed Configuration and Platform Implementation

To test the platform's performance and efficiency and also to examine its operation in a real environment, we deployed it in our experimental MANET testbed that comprises 4 PDAs and 2 laptops. Table 1 summarizes the configuration of the testbed. The testbed is a 6-hop MANET and is considered a relatively reliable environment. Relevant results on fundamental parameters can also be used as input to the simulations and the analytical modeling so that results can be extrapolated and general conclusions can be drawn. The performance metrics we measured were the overall control cost incurred by our design and the convergence time for a full system cycle as described in section III-C.

The communication between mobile nodes uses the lightweight XML-RPC protocol [13]. XML-RPC is a subset of the Simple Object Access Protocol (SOAP) with only basic functionality supported. 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

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 802.11b

TABLE 1. *Experimental testbed configuration.*

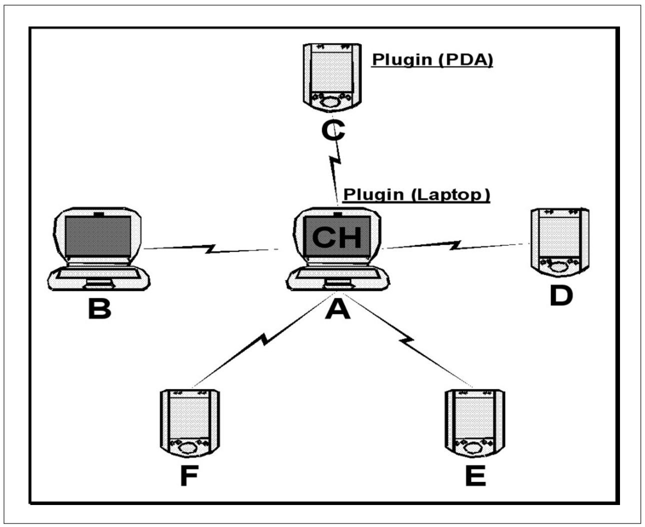

FIGURE 5. *Star topology.*

the RPC protocol itself. We chose an XML-based approach because we also use XML to represent contextual data collected by mobile nodes. We could have possibly chosen other distributed object technologies such as CORBA, or even full-fledged Web Services, but these technologies necessitate object interface/service endpoint advertisement and discovery functionality. In a volatile environment such as a MANET, discovery functionality adds one more level of complexity. In addition, the simple, static interfaces of our programmable platform and relevant interactions can be modeled by message passing through RPCs. Given our performance evaluation of XML and other management technologies [3], we believe that XML-RPC provides a useful blend of functionality and performance.

Trivial FTP (TFTP) [16] was used for the distribution of the plugins. It is less complex than FTP and has no user authentication, being mostly targeted to software-driven automated software download/upload. This saves both time and traffic in a trusted environment as the one we assume in our case, where trust relationships among nodes will be established. TFTP also uses only one connection, contrary to FTP that requires two connections for control and data traffic.

The platform is implemented using the Java 2 Micro Edition (J2ME) [17]. This version requires a small memory footprint, while at the same time it is optimized for the processing power and I/O capabilities of small mobile devices. We chose to use Java because of its ubiquity and platform independence. The use of Java requires nodes to have a Java Runtime Environment (JRE) installed. Although this is relatively memory-hungry, our hands-on experience confirms that even the resource-constrained PDAs can comfortably support its execution.

B. Testbed Experimentation

Our experimental testbed was used to test and evaluate the programmable platform in a real ad hoc network environment. The scenario implemented was that of the dynamic routing protocol change according to node mobility context information. As described earlier, nodes monitor their own mobility and report this to the CH/NH, which is responsible for identifying the need to potentially switch to a different routing strategy. The two routing protocols we used (reactive AODV-UU and proactive OLSR) are realized as C-based user space daemons. Practical problems encountered during the experiments included wireless link interference given that the wireless interfaces were deployed in a confined space. These interferences did not influence the experiment results but necessitated numerous executions of the experiments. In addition, since testing various network topologies was necessary, we used a MAC address filter tool to emulate broken links or unreachable destinations due to node mobility.

We experimented with many different topologies, routing protocols and other plugins to evaluate the platform's operation. In the following subsections, we present experimental results regarding the routing protocol switch scenario for different but representative network topologies, such as *star* and *line/bus*, but also for *full random mobility* scenarios. The star topology models a centralized approach, with the CH/NH conveniently located in the center, having a 1-hop distance from all other nodes. The line topology is the one that performs worse since it models a sparse MANET with 5-hop diameter. For the full mobility scenarios, bearing in mind the practical problems arising from physically moving around our experimental testbed, we decided to use a custom-made emulator to imitate actual movements. Our emulator is based on the principles in [21], is realized in Java so as to be easily integrated in our platform and makes nodes appear as moving around with links breaking and being re-established, using the aforementioned MAC address filter tool. The nodes are moving based on the mobility model described previously.

We have also implemented context gathering, processing and dissemination, although we do not describe relevant aspects here. In this scenario however, getting mobility information requires sensors such as accelerometers, GPS support, etc. For practical reasons we used emulated versions of these sensors. Relevant emulated mobility information is passed to the CH/NH to decide upon routing protocol switching. In this work we are mostly interested to assess the performance of our approach and platform in terms of plugin distribution and activation. For further details on our work on context-related issues, please refer to [22]. We should finally mention that the results were produced through a number of identical experiments and mean values are presented.

For comparison with the results presented next, we also measured the convergence time and management traffic for the ideal situation in which all nodes have both plugins, i.e. AODV and OLSR. In this situation plugin distribution is not required and this leads to minimized latency and control overhead that represents an ideal lower bound. These experiments were performed for a random topology and the convergence time was 10.82 seconds and the management traffic 48523 bytes on average.

1) Star Topology — The star topology deployed in our testbed is depicted in Figure 5. Laptop A was chosen as the center of the star and the CH/NH. Note that given the fact that the actual network is relatively small in terms of number of nodes and geographical area covered, it comprises a single cluster. The plugin to be elected and distributed (the C-based OLSR routing daemon) exists initially in two nodes, namely A and C. These two versions are different, reflecting platform-specific implementations.

The size of the plugin is 98.1 Kbytes for the PDA version and 454 Kbytes for the laptop one and is distributed to node B from node A and to nodes D, E, F from node C. All the traffic is routed through node A, which is the CH. The results taken from measurements for the complete system execution are shown in Table 2. The overall convergence time for the complete routing protocol deployment process (including an average time of ~4 seconds to initialize the new routing protocol) in the considered network topology is 25.43 seconds. This relatively long latency is attributed to the following two reasons: i) in the topology considered, only two nodes have initially the plugin, and two different versions need to be disseminated, ii) the size of the routing protocol plugin, which is

Metric	Value
Time convergence	25.43 sec
Routing traffic	7736 bytes
Control overhead	41742 bytes
TFTP traffic	1064880 bytes

TABLE 2. *Star topology experimental results.*

relatively big. These facts together with the underlying DCF-based IEEE 802.11 MAC lead to increased contention with an increase in binary exponential backoff, which in turn increases the convergence time. The PDA version of the plugin is transferred over 6 hops in total and the laptop plugin over 1 hop. The routing related traffic includes mainly "HELLO" messages used to ascertain route validity and was measured to be 7736 bytes. The overall control traffic overhead (41742 bytes) can be divided into CH-related traffic and non CH-related traffic. The CH traffic (8661 bytes) refers to the election-related traffic (request for advertisements and advertisement of plugins), the triggering of the distribution to the plugin owners, the requests and replies to locate plugin owners of a particular platform and the plugin installation confirmation by the nodes. The non-CH traffic (33081 bytes) includes transmissions incurred for requesting the plugin advertisement, the querying of neighboring nodes to check for the presence and the distribution and activation of the plugin. The plugin distribution traffic, caused by the TFTP file transfers was measured to be 1064880 bytes overall. This significant traffic is justified if one considers the size of the distributed plugin and the number of hops it is required to traverse (6 hops in total for the PDA plugin and 1 hop for the laptop one).

2) Line (bus) Topology — The line (bus) topology deployed in our testbed is depicted in Figure 6, with a laptop (A) chosen to be the CH/NH. The purpose of this experiment is to assess the operation of our platform in a relatively large-scale real MANETs, where distances between any node pairs tend to be high. The longest path in this experiment consists of five hops, which in MANETs is a relatively long distance. The plugin to be elected and distributed exists initially in two nodes, namely A and C. Note that the CH is located at the "edge" of the line topology, and the PDA with plugin one hop from the end, so that the plugin dissemination distances are as high as possible. The experimentation parameters remain the same as in the previous test cases. The results as measured during multiple executions of the scenario are presented in Table 3.

The time required for a complete system convergence was measured to be 31.74 seconds. The increase in time compared to the previous scenario is reasonable considering the significant increase in network diameter. This is also the reason for the increased routing related traffic (14332 bytes), since longer routes must be maintained. The control traffic has also increased (83145 bytes) since the CH is located at the network edge and the non-CH traffic also increased, considering the increased distance among nodes. The TFTP traffic is again significant but is mostly attributed to the large laptop plugin being transmitted over two links, while the PDA plugin is transmitted over 4 links.

3) Full Random Mobility Scenarios — Full random mobility scenarios were also considered by exploiting our custom built MANET emulator. We created a multitude of mobility trace files, based on the mobility model described in section IV, and subsequently experimented with our platform. Nodes A and C were always selected as the routing plugin owners, their positioning though was obviously not always the same. The measurements displayed in Table 4 represent average values gathered under various topology and mobility scenarios. The benefit of the emulation is that we exploit real MANET characteristics and evaluate our platform under various mobility scenarios. In our experiments, we emulated mobility based on scenarios in which we varied the terrain size from 100×100m to 200x200m since we only had 6 nodes and larger terrains would lead to continuous breaks of communication links. The relative node velocity was varied from 5 m/s to 20 m/s.

The first observation is that in these cases the convergence time is smaller than the previous static cases (22.57 to 28.07 seconds). The reason for this is that due to the limited terrain size and the fact that the range of wireless transmissions is 100m, there is increased connectivity between the nodes. Link and connectivity changes and disruptions caused by increasing node velocity yield a relative increase in traffic measurements (42092 to 55259 bytes) and convergence time. The routing traffic is considerably bigger in the case of the larger terrain size since more link breaks cause increased number of requests for route reconstruction. The TFTP traffic in the 100×100 terrain size scenario is much less than in any other case, since under these conditions all nodes are connected via direct links to each other and so the plugin acquisition is immediate. Our platform is robust enough to cater for link breaks by assuming a threshold value for the number of nodes that need to be present in order to complete the operation, as previously described.

The extensive testbed experimentation presented and discussed above first validates that our proposed framework operates properly in a real MANET environment. It also illustrates that our framework does not cause significant overhead in terms both of control traffic and convergence time. The maximum control traffic observed was 83145 bytes, which is distributed among 6 nodes that communicate in the worst case through 2Mb/s wireless links; this control overhead is deemed acceptable. Convergence time was measured to be 24 seconds on average. This time might seem significant but one needs to take into account that the particular experiments distribute a

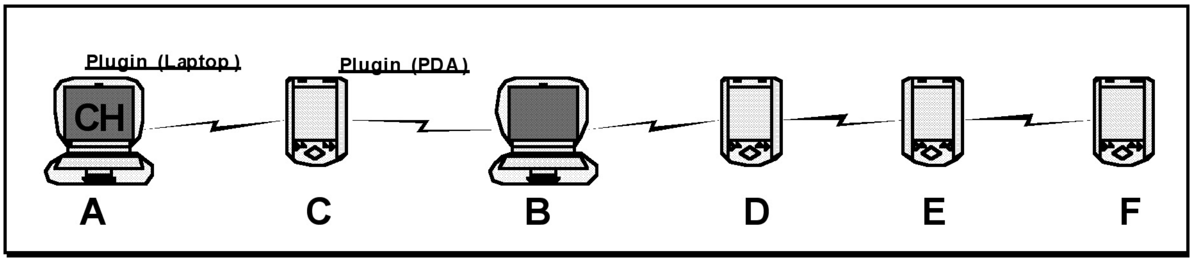

FIGURE 6. *Line (bus) topology.*

Metric	Value
Time convergence	31.74 sec
Routing traffic	14332 bytes
Control overhead	83145 bytes
TFTP traffic	1530924 bytes

TABLE 3. *Line topology experiment results*

454 Kbyte plugin across many hops. In addition, an average time of 4 seconds is necessary to initiate the new routing protocol and this is part of the convergence time, which is overall considered as satisfactory.

C. Simulation Analysis

The testbed experimentation validated the operation of our platform in a real-world scenario of a small-scale MANET. On the other hand, the objective of the simulation is to validate testbed experimentation but also to investigate the actual performance of our programmable framework in a large-scale MANET. Performance is analyzed in similar manner as before i.e. in terms of control traffic overhead and overall convergence time

We performed our simulations using GloMoSim in which we implemented the associativity-based CH/NH election heuristic [2] — note that in the testbed experiments we hard-assigned the CH/NH role — and the dynamic routing protocol switching, including plugin election and dissemination. The transmission range of each node is set to 100m, and the link capacity is considered to be 2 Mb/s (worst-case scenario). As described earlier, each CH/NH switches between reactive (i.e. AODV) and proactive (i.e. OLSR) routing strategies based on context information. In our simulations the context-driven need for routing protocol switching is derived based on the node mobility patterns. Group mobility is derived from individual node mobility patterns, using the same settings, i.e. mobility traces, as those used for testbed experimentation. Plugin switching latency is set to be 4 seconds as measured on average in the testbed experiments.

For the first stage of our simulation analysis that was presented in more detail in [18], we attempt to investigate the performance of our framework in terms of the average control traffic incurred per node and the convergence time required for the complete plugin deployment and activation. In order to assess the effect of increasing network size on the clustering and plugin election and distribution schemes, the terrain-area is also increased with an increase in the number of nodes, so that the average node density is kept constant. The number of nodes in this case is varied from 5, 20, 45, 80 and 125. The terrain-area size is varied so that the average node degree remains the same and accordingly 200×200 m^2, 400×400 m^2, 600×600 m^2, 800×800 m^2 and 1000×1000 m^2 are selected for each scenario (each cluster has a dimension of 200×200 m^2).

Figure 7 shows the average control traffic per node because of both clustering and programmable platform interactions; this is shown as a function of increasing number of nodes. The clustering-related control traffic of CH/NH election process is actually the traffic involved due to HELLO packet transmissions, and the control traffic associated with our programmable framework is the total traffic involved for the entire plugin deployment process as described in section III-C (excluding plugin distribution traffic). It can be inferred from Figure 7 that the average control traffic does not depend on the increasing node count, and hence both the clustering and plugin distribution schemes are scalable. The initial peak in the plugin-related control traffic is attributed to the small number of nodes, which leads to frequent connectivity losses in the designated terrain area.

It is evident that the control overhead measured through simulation is in the same range (approximately 6* 9100 = 54600 bytes) as the overhead measured during testbed experimentation, validating thus our approach.

Figure 8 depicts the average convergence time per node for complete plugin deployment as a function of increasing number of nodes. The convergence time is actually the time the plugin process takes from the point when a Terminode receives the plugin election trigger message from the CH until it finally receives the plugin activation message. It is evident from Figure 8 that the convergence time increases almost linearly with the node count. This was expected as we assume that the number of plugin owner nodes also increases proportionally with network size. Although these nodes are randomly distributed, in our simulation scenarios they appear to be reasonably well distributed as the network grows in size, avoiding "empty" areas, hence the almost linear convergence time with the number of nodes.

If we consider a population of 6 nodes as it was the case in testbed experiments, the overall convergence time measured through simulations is similar to that observed in the testbed. The average convergence time per node is around 4.5 seconds (first point in the graph of Figure 8) and the overall convergence time is 6*4.5=27 seconds, which is very similar to the one measured in the testbed experiments (see terrain size 200x200 on Table 4); this further validates our approach.

D. Analytical Modeling

Testbed experimentation assessed the performance of our platform in a real environment and more importantly provided concrete proof that it can provide the desired functionality, imposing a relatively small control overhead and achieving acceptable convergence times, in the order of 4 seconds per

Metric	Value		
Node relative velocity	5 m/s	10 m/s	20 m/s
(Terrain size: 100 × 100)			
Time convergence (sec)	22.57	23.14	23.81
Routing traffic (bytes)	7523	7645	7701
Control overhead (bytes)	42092	42816	43902
TFTP traffic (bytes)	901145	902123	902556
(Terrain size: 200 × 200)			
Time convergence (sec)	27	27.43	28.07
Routing traffic (bytes)	10209	12788	13342
Control overhead (bytes)	50512	52813	55259
TFTP traffic (bytes)	1101232	1104776	1105098

TABLE 4. *Full random mobility experiment results.*

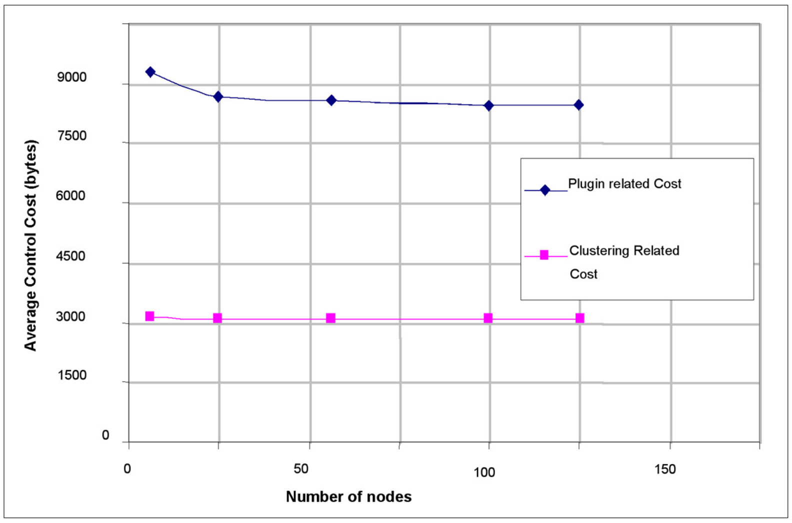

FIGURE 7. *Average control traffic per node (clustering and plugin deployment) vs. increasing node count.*

node for a network of 6 nodes i.e. ~24 seconds in total. Simulation experiments proved the scalability of our approach in large-scale MANETs and validated the results obtained from the practical experiments. In this section we attempt to model analytically the efficiency of plugin distribution in our platform and, based on this, validate our testbed experimentation and simulation results. In order to perform this probabilistic efficiency analysis, we define a set of evaluation metrics:

- *Average and Overall Control Overhead* (ACO and OCO): control traffic of exchanged messages for alignment to be achieved
- *Average number of hops to acquire plugin* ($\overline{K}$)
- *Probability of a node acquiring the plugin in* $\overline{K}$ *hops* ($\mathbf{P}_{\text{ACCESS}}$)
- *Probability of all nodes eventually acquiring the plugin in* $\overline{K}$ *hops* ($\mathbf{P}_{\text{ALIGNMENT}}$)

The general case allowing for full connectivity among all nodes is extremely difficult to address given the space limitations of this paper. For simplicity, yet without loss of generality, we make certain assumptions in order to reduce the problem complexity. In the absence of any power-control mechanism, the number of neighbors needed to maintain connectivity in the considered MANET has been proven to be $\Theta(\log N)$ [21]. We assume this number of neighbors, so connectivity is not an issue in our analysis. We also assume that all nodes are of the same platform, so different plugin versions are not considered.

Let N be the number of nodes in the MANET. Multihop communication can occur in K_{MAX} hops at maximum. The maximum hop count is N–1 in the extreme case that all nodes form a fully linear topology. We acknowledge that the maximum hop count is also restricted by the TTL values of IP and that of the routing protocol; for the purpose of our analysis though we consider that these TTL values can be set to the conceptual maximum of N–1. At a particular time t, the MANET nodes that possess the elected plugin are $N_P(t)$, $N_P(t) \le N$. By $Nb_i(x)$ or, by simply Nb_i, we denote the set of neighbor nodes to node x that are i hops away. It is obvious that when i = 0, Nb_0 = 1.

We denote as $p_i(x)$ or simply p_i the probability of a node to acquire the requested plugin in i hops and no less. We make the plausible assumption that plugin owners are uniformly distributed throughout the network. This means that the number of plugin owners in the i neighborhood of a node is $(Nb_i * N_p)/N$. Under this assumption, every node has an equal probability of being a plugin owner. In this respect the probability of a node having the plugin itself, defined as p_0, is

$$p_0 = \frac{N_P}{N}$$

The probability of a node not having the plugin is $1 - p_0$. The probability p_1 of a node acquiring the plugin in its 1-hop neighborhood, assuming the node itself is not a plugin owner, is given by the following equation that combines these two events, namely the node not having the plugin and at least one of its 1-hop neighbors having it:

$$p_1 = (1-p_0) * \left(1-(1-p_0)^{Nb_1}\right)$$

Based on the previous equations, the probability of a node acquiring the desired plugin from a node in its K neighborhood is p_K and this is actually the P_{ACCESS} metric we defined earlier when the number of hops is averaged. This is the combined probability of finding the plugin in at least one of the K hops neighbors and not having found it up to the K – 1 hop neighborhood:

$$p_K = (1-p_0)^{1+Nb_1+K+Nb_{K-1}} * \left(1-(1-p_0)^{Nb_K}\right)$$

The first part of the formula denotes the probability of no other node in less than K hops having the plugin. The second part of the formula refers to the probability of at least one node in the K hop neighborhood being a plugin owner.

A useful metric is the average number of hops required by a node to successfully acquire the plugin. We denote as $\overline{K}$ this average hop count of finding a plugin owner and it can be

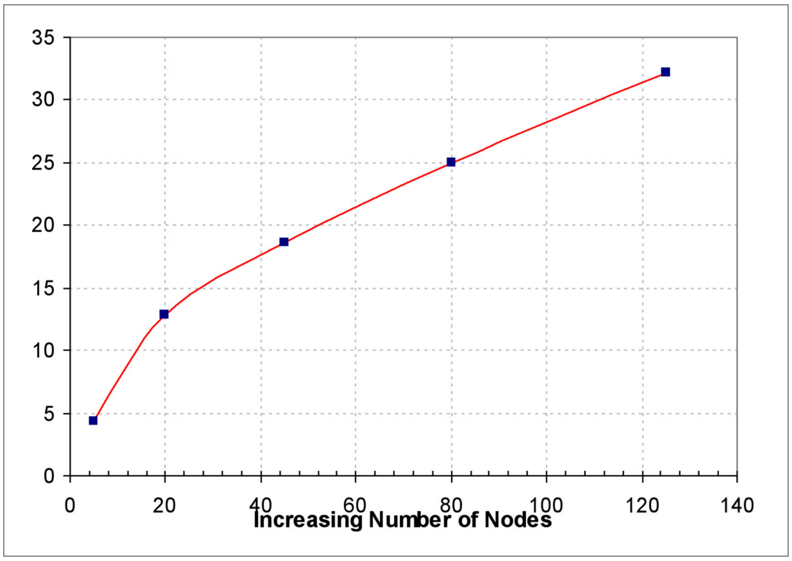

FIGURE 8. *Average per node convergence time for the plugin deployment process vs. increasing node count.*

easily seen that.

$$\overline{K} = \frac{\sum_{j=1}^{N} \sum_{i=1}^{N-1} i * p_i(j)}{N - N_P}.$$

This formula denotes that the average hop count is given from the sum of all possible hop counts (up to the maximum of $N - 1$) multiplied by the probability to have a plugin owner at that hop count. This is averaged over the number of nodes in the MANET that do not have the plugin.

Based on the average hop count ($\overline{K}$) a node requires to get the plugin, we can calculate the average number of messages that need to be sent over the MANET for a node to acquire the plugin in $\overline{K}$ hops. Assuming the overhead M of messages exchanged per hop as constant, the average control overhead is $ACO = \overline{K} * M$, hence the overall control overhead for all nodes of the MANET is given by:

$$OCO = ACO * (N - N_p) = \overline{K} * M * (N - N_p)$$

The probability that a node will find a plugin owner in $\overline{K}$ hops and thus obtain the required plugin is P_{ACCESS}. Based on the previous formulae, this probability is given by the following equation:

$$P_{ACCESS} = (1 - p_0)^{1+Nb_1+\ldots+Nb_{\overline{K}-1}} * \left(1 - (1 - p_0)^{Nb_{\overline{K}}}\right)$$

The combined probability for every node to acquire the plugin in an average distance of $\overline{K}$ hops, which we call $P_{\text{ALIGNMENT}}$ as it reflects full network alignment, is given by

$$P_{ALIGNMENT} = P_{ACCESS}^{N-N_P}$$

All previous calculations assumed a uniform distribution of plugin owners throughout the MANET. It could be possible to cater for other distributions, but this would complicate the presented analysis. In short, in this case the probability of plugin distribution in a 1-hop neighborhood would be defined in accordance to the selected distribution and the analysis would have to adhere to the new definition, starting from p_0.

We described the operation of our platform earlier. Based on this description, we attempt to calculate the average control overhead per hop imposed by our platform (defined previously as M) during a full cycle of system operation. If we disregard the CH-related traffic that involves plugin election, then the traffic generated per hop is derived from the following series of XML-RPC calls. A Terminode floods the advertisement of its plugins (advertisePlugin call, measured in the testbed to be 1211 bytes on average), then queries the neighbors on whether they own the plugin and waits for their reply (hasPlugin call and reply with approximate traffic of 2*1098 bytes). The Terminode then invokes the receivePlugin call to handle receiving and installation of the plugin (1211 bytes) and finally handles the activatePlugin call that deals with local plugin activation and informs the CH of the successful alignment (1152 bytes). The CH-related traffic involves the request for plugin election (approximately 1084 bytes), the request to the elected plugin owner to distribute the plugin (on average 1095 bytes) and the confirmation of plugin activation (524 bytes). Thus the overall number of bytes on average is M = 8473 bytes.

Having defined the average per hop overhead imposed by our framework, we can calculate the Overall Control Overhead for various node populations given the formula we derived previously. For the testbed experiments we have N = 6, N_P = 2 and consequently the average number of neighbors of a node is log$N$ = 2.5. Using these values the average hop count for our testbed experiments is ~1.5 as it can be calculated from the corresponding formula. We do not provide relevant calculation details due to space limitations. Consequently, the analytically derived formula gives overall control overhead OCO=1.5*8473*(6–2) = 50838 bytes. This value is very similar to the values measured both through testbed experimentation (see terrain size 200 × 200 on Table 4) and simulation (9100*6 = ~54600 bytes — first point in the graph of Fig. 7), validating thus further our approach.

VI. Conclusions and Future Work

In this paper we presented a programmable middleware framework that can align the capabilities of the nodes of a MANET through the use of loadable plugins. Relevant alignment can be initiated either by explicit human user decisions, or in an adaptive dynamic manner based on context information. This achieves the alignment of capabilities in a heterogeneous environment such as a MANET and can be used to dynamically deploy protocols that may result in a more optimal MANET operation, as shown by our application scenario. It can also be used to deploy application servers at selected nodes so that most other nodes have relatively close access to them.

The platform has been implemented and evaluated in our experimental testbed, allowing us to get a better understanding of its operation in actual deployment scenarios. Although the proposed platform can support service and protocol deployment at any level, we presented an example scenario that addresses adaptive and dynamic network-level protocol deployment. This distinguishes our work from the majority of related work that focuses on application-level service deployment and provides evidence regarding the platform efficiency in supporting various configurations and applicability scenarios. In our application scenario, node mobility is measured and aggregated, characterizing the whole network as either highly volatile or relatively static. A decision is made accordingly to deploy and switch to a new, more suitable, MANET routing protocol on the fly when a particular network mobility threshold is crossed.

Our testbed experiments validated the correct platform operation and exhibited relatively good convergence times given that a relatively large size plugin was distributed i.e. a new routing protocol daemon. Although we chose dynamic routing protocol deployment in our initial case study, more complex services can be deployed using our framework. The cross-layer operation of our approach, in which application level context information and programmable infrastructure result in different network layer configuration and operation, demonstrates its usefulness, extensibility and strength.

Our experimental results have been assessed through testbed experimentation, simulation and analytical modeling, each assessment technique validating the others. This both ensured the reliability of our evaluation and also validated the correct operation of our platform according to its design specifications.

We have adopted a hierarchical management approach, with CHs administering clusters and one of them nominated as NH, administering the whole network. The approach could be centralized, with the NH taking all decisions, or partly distributed, with management decisions reached through collaboration among the CHs or a selected set of CHs.

Aspects we plan to work on include the following: evaluation of the context-management infrastructure which we have designed and implemented [22] but did not consider here in detail; security for distributing plugins in the ad hoc network and enabling trust among nodes; a peer-to-peer paradigm for collective decision making among CHs that belong to a MANET dominating set; and a policy-based model for the autonomic triggering of management decisions based on both predefined and on-the-fly introduced policies.

VII. Acknowledgments

The work presented in this paper was carried out in the context of the Programmable Ad hoc Networks (PAN) UK EPSRC project — GR/S02129/01.

References

[1] S. Sivavakeesar *et al.*, "Effective Management Through Prediction-Based Clustering Approach in the Next-Generation Ad Hoc Networks," *Proc. IEEE Int'l. Conf. Commun. (ICC'04)*, vol. 7, June 2004, pp. 4326–30.

[2] S. Sivavakeesar, and G. Pavlou, "Associativity-Based Stable Cluster Formation in Mobile Ad Hoc Networks," *Proc. IEEE Consumer Communications & Networking Conf. (CCNC 2005)*, Nevada, USA, Jan. 2005, pp. 196–201.

[3] G. Pavlou *et al.*, "On Management Technologies and the Potential of Web Services," *IEEE Commun. Mag.*, special issue on XML-based Management of Networks and Services, vol. 42, no. 7, IEEE, July 2004, pp. 58–66.

[4] D. Tennenhouse *et al.*, "A Survey of Active Network Research," *IEEE Commun. Mag.*, vol. 35, no. 1, Jan. 1997, pp. 80–86.

[5] C. Tschudin, H. Lundgren, and H. Gulbrandsen, "Active Routing for Ad hoc Networks," *IEEE Commun. Mag.*, vol. 38, no. 4, Apr. 2000, pp. 122–27.

[6] C. Bohoris, A. Liotta, and G. Pavlou, "Evaluation of Constrained Mobility for Programmability in Network Management," *Proc. 11th IEEE/IFIP Int'l. Wksp. Distributed Systems: Operations and Management (DSOM'00)*, Austin, Texas, USA, A. Ambler (Ed.), Springer, Dec. 2000, pp. 243–57.

[7] G. Goldszmidt and Y. Yemini, "Evaluating Management Decisions via Delegation," *Proc. IEEE Integrated Network Management III*, Elsevier, 1993, pp. 247–57.

[8] A. Lazar, "Programming Telecommunication Networks," *IEEE Network*, vol. 11, no. 5, Sept.–Oct. 1997, pp. 8–18.

[9] J. Biswas *et al.*, "The IEEE P1520 Standards Initiative for Programmable Network Interfaces," *IEEE Commun. Mag.*, vol. 36, no. 10, Oct. 1998, pp. 64–70.

[10] O. Angin *et al.*, "The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking," *IEEE Pers. Commun. Mag.*, vol. 5, no. 4, August 1998, pp. 32-43.

[11] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, Ad hoc On-Demand Distance Vector (AODV) Routing, IETF RFC 3561, July 2003.

[12] B. N. Schilit, D. M. Hilbert, and J. Trevor, "Context-Aware Communication," *IEEE Wireless Commun.*, vol. 9, no. 5, Oct. 2002, pp. 46–54.

[13] XML-RPC specifications web site, http://www.xmlrpc.com/spec, accessed Jan. 2007.

[14] B. Schwartz *et al.*, "Smart Packets: Applying Active Networks to Network Management," *ACM Trans. Computer Systems*, vol. 18, issue 1, Feb. 2000, pp. 67–88.

[15] A. Boulis and M. B. Srivastava, "A Framework for Efficient and Programmable Sensor Networks," *IEEE OpenArch 2002*, pp. 117–28.

[16] K. Sollins, The TFTP Protocol, IETF RFC 1350, July 1992.

[17] Sun J2ME Homepage, http://java.sun.com/j2me/index.jsp, accessed Jan. 2007.

[18] S. Gouveris *et al.*, "Programmable Middleware for the Dynamic Deployment of Services and Protocols in Ad hoc Networks," to appear in the *Proc. IEEE/IFIP Integrated Management Symposium (IM 2005)*, Nice, France, IEEE, May 2005, pp. 3–16.

[19] B. Li, "QoS-aware Adaptive Services in Mobile Ad-hoc Networks," *Proc. Int'l. Wksp. Quality of Service 2001 (IWQoS '01)*, Springer LNCS 2092, 2001, pp. 251–68.

[20] T. Clausen and P. Jacquet, Optimized Link State Routing Protocol (OLSR), RFC 3626, Oct. 2003.

[21] Y. Zhang and W. Li, "An Integrated Environment for Testing Mobile Ad-Hoc Networks," *ACM MobiHoc 2002*, Switzerland, June 2002, pp. 104–11.

[22] A. Malatras, A. Hadjiantonis, and G. Pavlou, "Exploiting Context-Awareness for the Autonomic Management of Mobile Ad Hoc Networks," *J. Network and Systems Management*, Springer, vol. 15, no. 1, Mar. 2007.

[23] V. Ramasubramanian, Z. Haas, and E. G. Sirer, "SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks," *ACM MobiHoc 2003*, Maryland USA, June 2003, pp. 303–14.

[24] M. Pearlman and Z. Haas, "Determining the Optimal Configuration of the Zone Routing Protocol," *IEEE JSAC*, vol. 17, no. 6, Aug. 1999, pp. 1395–414.

[25] A. Hadjiantonis, A. Malatras, and G. Pavlou, "A Context-Aware, Policy-Based Framework for the Management of MANETs," *7th IEEE Int'l. Wksp. Policies for Distributed Systems and Networks (POLICY '06)*, June 2006.

[26] H. Yang *et al.*, "Security in Mobile Ad Hoc Networks: Challenges and Solutions," *IEEE Wireless Commun.*, vol. 11, no. 1, Feb. 2004, pp. 38–47.

[27] Y. L. Sun *et al.*, "Information Theoretic Framework of Trust Modeling and Evaluation for Ad Hoc Networks," *IEEE JSAC*, vol. 24, no. 2, Feb. 2006, pp. 305–17.

[28] M. Bechler *et al.*, "A Cluster-Based Security Architecture for Ad Hoc Networks," *IEEE INFOCOM 2004*, vol. 4, Mar. 2004, pp. 2393–404.

Biographies

Apostolos Malatras (A.Malatras@surrey.ac.uk) is a post-doctoral researcher at the Centre for Communication Systems Research, Department of Electronic Engineering, University of Surrey, UK, where he is an active member of the Networks Research Group. He holds a Diploma in computer science from the University of Piraeus, Greece, a M.Sc. degree in Information Systems from the Athens University of Economics and Business, Greece, and a Ph.D. degree in Networking from the University of Surrey, UK. His research interests focus on context awareness, knowledge management, network management, and communications middleware.

George Pavlou (G.Pavlou@surrey.ac.uk) is professor of communication and information systems at the Centre for Communication Systems Research, Department of Electronic Engineering, University of Surrey, UK, where he leads the activities of the Networks Research Group. He holds a Diploma in engineering from the National Technical University of Athens, Greece, and M.Sc. and Ph.D. degrees in computer science from University College London, UK. His research interests focus on network management, networking, and service engineering, covering aspects such protocol performance evaluation, traffic engineering, quality of service management, policy-based systems, multimedia service control, programmable networks, and communications middleware.

Siva Sivavakeesar (S.Sivavakeesar@surrey.ac.uk) is post-doctoral researcher at the Centre for Communication Systems Research, Department of Electronic Engineering, University of Surrey, UK, where he is pursuing research in quality of service issues associated with ad hoc networks. He has been involved in the "Programmable Ad hoc Networks" project that considers ad-hoc networks as an important emerging type of network infrastructure that will complement fixed network infrastructures. His research interests focus on routing and quality of service for wireless ad hoc networks.