MIDDLEWARE SERVICES FOR INFORMATION SHARING IN MOBILE AD-HOC NETWORKS

Challenges and Approach

Thomas Plagemann[1], Jon Andersson[2], Ovidiu Drugan[1], Vera Goebel[1], Carsten Griwodz[1], Pål Halvorsen[1], Ellen Munthe-Kaas[1], Matija Puzar[1], Norun Sanderson[1], Katrine Stemland Skjelsvik[1]

[1]*University of Oslo, Department of Informatics, P.O. Box 1080, 0316 Oslo, Norway;*
[2]*Thales Communications AS, P.O. Box 6611 Etterstad, 0609 Oslo, Norway*

Abstract: Information sharing is a mission critical key element in rescue and emergency operations. Mobile ad-hoc networks (MANETs) could provide a useful infrastructure to support information sharing, but appropriate applications are needed. To facilitate efficient application development for this type of infrastructure, middleware support is needed. In the Ad-Hoc InfoWare project, we are currently developing corresponding middleware services. In this paper, we discuss the application requirements that are imposed onto the middleware services, and we outline our technical approach to address the corresponding challenges. The architecture we propose comprises five main building blocks, namely knowledge management, a local and a distributed event notification service, resource management, and security and privacy management. We indicate design alternatives for these building blocks, identify open problems and relate our approach to the state-of-the-art.

Key words: Middleware services, information sharing, mobile ad-hoc networks, knowledge management, event notification, resource management, security.

1. INTRODUCTION AND MOTIVATION

Efficient collaboration between rescue personnel from various organizations is a mission critical key element for a successful operation in emergency and rescue situations. There are two central preconditions for efficient collaboration, (1) the incentive to collaborate, which is naturally given for rescue personnel, and (2) the ability to efficiently communicate

and share information. Mobile ad-hoc networks (MANETs) could provide the technical platform for efficient information sharing in such scenarios, assuming that all rescue personnel is carrying and using mobile computing devices with wireless network interfaces. Applications are needed to turn a working infrastructure of a MANET into a useful system, like dispatching of rescue personnel and equipment, context-aware medical diagnosis and treatment support, and real-time evidence collection and management. However, application development for MANETs is not easy. MANETs are typically highly dynamic networks in terms of available communication partners, available network resources, connectivity, etc. Furthermore, the end-user devices are very heterogeneous, ranging from high-end laptops to low-end PDAs and mobile phones. CPU storage space, bandwidth, and battery power represent important resources. Finally, many application scenarios, like coordination of rescue teams, have also quite hard non-functional requirements, like availability (including reliability, fault tolerance, and survivability), efficient resource utilization, and security and privacy. Thus, sufficient quality in information access and sharing in such an environment is hurdled by quite many obstacles. Obviously, solving these issues in every new MANET application from scratch is not meaningful. Instead, a set of middleware services that support the development of applications for MANETs is needed.

Since the application domain of emergency and rescue scenarios differs from traditional application of MANETs, we regard it as important to identify the particular requirements of this applications scenario and their implications for the design of integrated middleware services. Therefore, we focus in this discussion paper on the requirements, the resulting challenges, and a description of our overall approach. In Section 2, we elaborate in more detail the requirements for middleware services that support information sharing in MANETS for emergency and rescue applications. Section 3 presents the blueprint of our approach. In Section 4, we conclude and describe future work.

2. APPLICATION SCENARIO AND APPLICATION REQUIREMENTS

It is our goal to develop middleware services for information sharing in emergency and rescue operations. We assume that wireless computing devices will be used as the basic technical means for information sharing between rescue personnel, like policemen, firemen, physicians, and paramedics. The number of devices present at an emergency site is probably not larger than hundred(s). Including small sensors that are either at the site

or introduced by rescue personnel, the number of devices may eventually reach up to (ten) thousands, but it will still be considerably smaller than Internet scale. These devices form MANETs at emergency sites with all their well known properties, like heterogeneous nodes, unpredictable reachability of nodes, etc. However, MANETs at emergency sites might not be entirely infrastructureless, because some devices might serve as gateways to the Internet. Another important difference to classical application scenarios for MANETs is the fact that certain preparations for rescue operations can be done in advance with full access to the Internet. In particular, we distinguish six phases in such a scenario: *A priori*, before the accident the different organizations will exchange information on data format, and make agreements on working methods. After an accident has happened, the first step is *briefing* of the different rescue teams, involving gathering of information about the disaster, e.g., weather, location, number of people involved, and facilities in the area. The next phase is the *bootstrap* of the network where events such as registration of nodes and electing leaders take place. During the *running* of the network different events may happen that will affect the middleware services: a node may join or leave the network, the network may be partitioned, and network partitions may be merged again. At the *end* of the rescue operation, all services must be terminated. *After* the rescue operation it could be useful to analyze resource use, user movements, how and what type of information is shared to gain knowledge for future situations.

In the a priori and briefing phases, the devices are connected to a stable infrastructure. It would be optimal if all relevant information could be uploaded during these phases on the devices that need it. However, this is generally impossible, because some information cannot be accessed by all organizations due to for example privacy concerns or there is just not enough time in the briefing phase to identify and upload the information. Examples for this type of information include security codes of doors at an emergency site, detailed building plans, specification of freight on a vehicle or in a storage, or medical records of persons that are known to be involved in the emergency. Furthermore, information generated at the emergency site during the operation, like sensor readings of room temperatures, information about how many injured persons have been detected at which location, readings of health monitors attached to injured persons, or information indicating the causes of an emergency situation, etc.

The middleware must support sharing of such information during the running phase. To accommodate the heterogeneity of organizations involved, it must present the information in a way that all organizations can understand. This implies supporting functionality akin to high-level distributed database system functionality, querying available information

and keeping track of what information is available in the network. Ontologies are a means of explicating semantic knowledge about the information. The middleware must account for different domain ontologies and standards that might be used by the organizations. A major challenge for knowledge management is to support information sharing across organizations such that they understand each other's structure and data descriptions.

Another set of requirements is concerned with controlled access to shared information which has to be addressed by security and privacy solutions. For example, passive bystanders like journalists should not get access to medical records, and aggressive bystanders like terrorists should not be able to alter or delete important data which might sabotage the rescue operation.

The likelihood of connection loss implies also that middleware services based on synchronous communication are not a good choice, because they are too vulnerable with respect to communication disruptions. The alternative to synchronous solutions is a distributed event notification system (DENS). Devices can lose contact to other mobile devices due to network partitioning or power drain, but groups that are portioned off from other parts of the MANET should function as good as possible. Therefore, replication is necessary to achieve the required level of availability. In order to make replication decisions that increase the availability and result in efficient resource utilization, it is important to keep track of resources.

Performance and efficient resource utilization are also important, but there is typically a trade-off between these two requirements and availability. There is no general solution for this trade-off and its resolution often depends on the particular application and even the particular emergency situation. Therefore, it is necessary to allow the application to define policies on how to handle these tradeoffs. The heterogeneity of hardware requires also that middleware services are configurable such that small resource-weak devices run only a few middleware components and devices with sufficient resources run many (or all) components.

3. BUILDING BLOCKS

We address these challenges and requirements in the Ad-Hoc InfoWare project by developing a set of configurable middleware components for MANETs that provide their services to applications and to other middleware components. Figure 1 illustrates our architecture, comprising five major components and some sub-components: *knowledge management* to handle ontologies, metadata management, integration of metadata and information from different sources; two components for event notification, *distributed*

event notification to decouple subscribers from publishers through mediating nodes and *watchdogs* to notify about local events, *resource management* to keep track of neighboring nodes and their resources to provide information for replication decisions, and *security and privacy management* based on a priori gathered certificates, key management for signing and encrypting of messages, and access control.

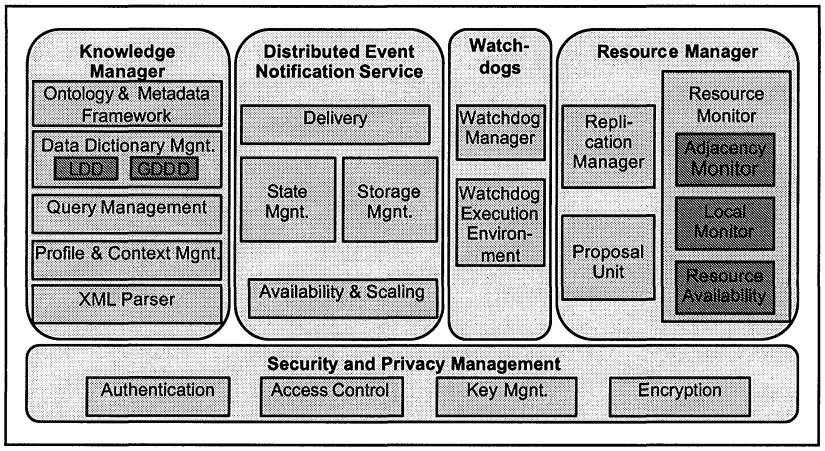

Figure 1. Main middleware components

3.1 Knowledge Manager

The purpose of this component is to manage knowledge sharing and integration in the network, by providing services which allow relating the metadata descriptions of the information items to a semantic context (through ontologies), and thus adding a layer of knowledge to the information shared in the MANET. One of the main tasks for the knowledge manager (KM) is to provide a framework for storage and management of metadata and ontologies for applications and middleware components. Additionally, it should manage storage of metadata, both content descriptions and schema/structure, and enable querying and retrieval of relevant information items and resources in the current network. It should also keep track of the availability of these. Another main task is to support the understanding of metadata from the participating domains to create a common pool of knowledge for the particular rescue operation. This will include some kind of mapping, merging, or integration of the knowledge each organization chooses to share. Therefore, the KM will offer the

following high-level services: data dictionary services, metadata and ontology framework services, profile and context services, and query services. In addition, it will provide an XML parser.

Global distributed data dictionaries (GDDD) will be used to provide a global view of what information is available in the network, and local data dictionaries (LDD) to view metadata of local information objects that can be shared with other nodes. Creating a GDDD simply by keeping copies of the content of the LDDs is not feasible for several reasons, among which are scarce resources, availability, and lack of a semantic context. A possible solution towards alleviating some of these problems is to create a GDDD by linking the content of the LDDs to a semantic context, which can be traversed and queried, like a semantic net (or web). A GDDD may request other nodes in its network range for relevant LDD content and subscribe to changes on this metadata by using the event notification service. The GDDDs also keep track of the availability of the items described in the LDD. This can be done in a pro-active (eager) or re-active (lazy) fashion, possibly depending on the current configuration.

Our focus for the KM is to facilitate use and sharing of existing domain ontologies from the participating organizations, and how to make these ontologies and metadata "understood" by all involved parties, so information can be shared across domains for the current application scenario. This can be solved by using an upper ontology to bridge the different domain ontologies. Each node has a set of resources, and may offer services to other nodes. Device profiles may contain this kind of information, as well as which context (e.g. time, location, and situation) the device currently has. User profiles may also show which role the user has in the current context, e.g., in a rescue scenario, we will most likely have team leaders, leader of communication, transport, rescue site leader and so on. The use of profiles and context enable personalization of data, support information filtering to avoid the overflow of irrelevant information, and allow relating information to a user's or a node's context. The KM will support management of profiles and context, so applications can create and manage profiles and context as needed. To allow a user or application to actively request data retrieval, the query manager should support different approaches, like naming of an information object, e.g., a URL, formulating a query, and filtering and/or ranking of the retrieved information according to context and profile.

Important related work for the KM includes MoGatu[14], Shark[15], and AmbientDB[8]. MoGatu is a framework for profile driven data management in a mobile ad-hoc environment using data-based routing, semantic-based data caching, and replication algorithms. It supports a point-to-point pull model. Shark is a system for organization, synchronization and exchange of knowledge among mobile users from one group and from different groups

by the use of knowledge ports declaring topics for knowledge exchange. The architecture relies on stationary server nodes. AmbientDB adds high-level data management functionalities to a distributed middleware layer by providing a global database abstraction over a MANET using Distributed Hash Tables. Our approach does not support a fully distributed database across all applications, but we may learn from their way of organizing metadata.

3.2 Event Notification

The distributed event notification service (DENS) comprises three delivery components to exchange information on subscriptions and notifications between the following three pairs of entities: subscriber – DENS, DENS – DENS, and DENS – publisher. At least one of these delivery components is needed by any node that wants to use and or provide DENS related services. The DENS itself consists of three management components, i.e., state management, storage management, and availability and scaling management. In order to detect local events, it uses watchdog (WD) management and WD execution environment with the resource manager. The subscription model the DENS provides is determined by the fact that any kind of data that might be stored in data structures in main memory, a file, or system internal tables could be of interest for subscribers. Therefore, a publisher cannot decide which information should be published. Instead, subscribers have to specify in which data they are interested. Thus, a kind of content-based subscription model is needed. This is realized through the concept of WDs. A WD is an agent whose task is to monitor within the node whether the condition that is specified in a subscription is fulfilled. In this case, it notifies the DENS that the event has occurred. The WD management allows starting and stopping WDs, and maintains a list of the WDs that are currently running on the node in the WD execution environment. All nodes that are willing to serve as publisher in the ad-hoc network have to implement this component. WDs can be used by the DENS and any other local process.

The DENS probably runs on mobile devices which might lose contact to other mobile devices due to network partitioning or might even be switched off to save power. In case of network partitioning, the DENS design supports information sharing in the different network partitions, and if arbitrary devices are switched off, including DENS nodes, it provides services in a best-effort manner. DENS nodes have to maintain the state information about subscriptions, i.e., the list of subscriptions, such that the corresponding notifications can be sent to the proper subscribers. To obtain highly available DENS, the state information is replicated among the nodes running DENS.

Network partitioning and network merging can easily lead to an inconsistent state. Therefore, the DENS is able to handle subscriptions and notifications with inconsistent state information and it includes maintenance protocols to obtain again a consistent state after these events. The storage manager stores information about notifications that could not be delivered. This allows to implement a delivery semantics that comes as close as possible to *at-least-once.*

One important difference between our approach and related work is that DENS components may be implemented on mobile nodes. STEAM[12] is an event-based middleware service tailored for ad-hoc networks. There is no intermediate middleware; instead a publisher will send notifications directly to its subscribers in the proximity. Siena[3] and JEDI[5] have support for mobile clients, but the clients, i.e., the publishers and subscribers, have the responsibility of telling the service that they have moved. Rebeca[7] uses virtual clients and pre-subscriptions to manage mobility and location-dependent subscriptions that are replicated to the virtual clients.

3.3 Resource Manager

The Resource Manager (RM) is a distributed service that manages information about resources, like available physical resources and software registered as resource, as well as reachability and relative positioning of nodes. The RM provides services to local applications and middleware components. The local information is controlled by the RM on the node, and it communicates with RMs on other nodes to disseminate information about remote resources as well. This division is necessary because of the unstable nature of MANETs, and it requires that at least a minimal configuration of the RM is available on each node. The internal structure of the RM comprises the three components Resource Monitor (ResMo), Proposal Unit (ProU) and Replication Manager (RepMng).

ResMo can be accessed by processes on the local node via a synchronous interface, which is called inspector, and an asynchronous interface called watcher. Inspectors deliver information in response to a query received by the RM. Watchers represent a hook for watchdogs on resources. Internally, ResMo builds on a local monitor, adjacency monitor, and a resource availability discovery unit. The three elements are concerned with gathering and maintaining information about local resources, links to direct neighbors and resources on other nodes, respectively. The first two work only locally and are therefore mandatory. The information about local resource is frequently updated. The third element may handle information from many nodes and is therefore optional. Depending on its use, the information is updated by notification whenever local resource information has changed or

on demand. The ProU, which is also optional, uses it to access information about remote nodes. A history of resource information must be provided to the ProU by the ResMo because it is required for predictions, when other components or applications inquire about the probability of network partitioning or information about the presence of certain resources. ProU uses also information managed by the KM from device profile and user profile. By using profile information from KM, ProU can define sets of nodes, referred to as groups, e.g., to limit the search for storage space for data belonging to a medical team to the teams own nodes. It can also distinguish among the group members, e.g., a node is used by a team leader which implies that it has the highest priority for receiving updates. The optional component RepMng uses the ProUs predictions for data replication to increase the availability in the network. The RepMng can be used to replicate, (1) the internal data of the local RM and (2) data for other processes. An example for the latter could be a user pointing out a large file and requesting two replicas on nodes that will be in close range in the future with a very high probability.

Although research in this area has been performed, the existing systems cannot be used directly in our scenario. For example, Chen et al.[4] and Li and Wang[11] propose systems which base their resource management on the ability to predict a possible partitioning of the network. These systems assume location services, e.g. GPS, which cannot be assumed in our scenario (e.g., inside a tunnel) or on all devices. Additionally, not all the nodes are able to predict a possible partition, nor to take part in the replication. We intend to integrate and adapt in our research interesting ideas from existing work, i.e., partitioning prediction based on movement patterns, and data replication which takes advantage of group partitioning prediction and replicates accordingly.

3.4 Security Manager

The possible security and privacy attacks can be roughly divided into two groups, external and internal[10]. External attacks include jamming, traffic congestion, incorrect routing messages, repeating messages, eavesdropping, impersonating, message manipulation, etc. Most of these problems can be solved relatively easily by means of standard encryption and digital signature techniques.* Internal attacks coming from nodes that have previ-

* Denial of service attacks on lower layers causing battery drain or network congestion, like cannot be handled at the middleware layer and as such are out of the scope of this work.

ously been authenticated, but later either lost or stolen, are harder to detect and a much bigger threat.

We have to distinguish authorized nodes, i.e., members of rescue organizations, from foreign nodes. Although it is not typical for pure ad-hoc networks, we use predefined information in the *a priori* phase to achieve that level of trust. One approach is to use a public key infrastructure (PKI), with a common certificate authority (CA) at the top, whose signature can then be verified by everyone.

The first security barrier encountered by an incoming message is the authentication barrier, located between the data link (MAC) layer and the network (IP) layer. This is mostly important for protecting the routing protocol[10], since incorrect routing messages could cause the network to function improperly, or not to function at all. All the messages coming to a node should be properly signed using a shared network key. When the first two nodes start bootstrapping the network, they first authenticate each other using preinstalled certificates, establish a secure channel, and create the network key. There is a high probability that several networks with different network keys will be formed, especially during the bootstrap procedure. When these networks try to merge, there is obviously a problem of network key inconsistency. Since there could be simultaneously more than one point of merging, there should be a non-ambiguous way of deciding which of the keys is "better". Ideally, the criteria would involve the number of nodes in each area, selecting the key which would cause a smaller number of nodes changing it. However, if this information is inconsistent, it could cause a key-exchange loop and thus introduce more harm than gain. Another, simpler approach is to choose the key with a lower ID, timestamp, etc. This avoids key-exchange loops, but a massive re-keying in a bigger area could be caused by the key coming from a much smaller area. After the nodes agree on a common key, the node making a change has to distribute the new key within its network area. This can be done proactively by some means of flooding, reactively, i.e., on demand, when a node detects traffic signed with the old key, or using a combination of the two. Solutions for these problems are currently designed and will soon be simulated and implemented within a common key-management algorithm. Most of the current key-exchange solutions are based on Diffie-Hellman key exchange and assume constant rekeying when nodes join and leave, as well as some kind of hierarchy[1, 2, 6]. Due to the high dynamics and probably scarce resources, these approaches might not be suited for managing the network key. However, they might be very useful for creation of dynamic groups or teams.

A problem which always emerges when introducing security is user friendliness. Security should be automated and transparent to users as much as possible, especially rescue operations, where human lives are involved

and there is no time to think about, i.e., synchronizing network keys. Other open issues include: data confidentiality, user authentication, encryption and digital signature algorithms, key-update mechanisms, protection from repeating messages, mechanisms for access control to information and resources, and choosing the right key to protect data, protection from lost or stolen nodes, etc. Revocation of certificates from lost or stolen nodes might also be a problem due to the high dynamics and lack of infrastructure. Another approach, if IPv6 is used as the network protocol, could be to cryptographically bind a node's IP address to its certificate [13], which would prevent nodes from using other nodes' IP addresses and therefore might allow an easy and efficient way to perform blacklisting of problematic nodes.

4. CONCLUSION AND FUTURE WORK

By analyzing the current state of research in MANETs and applications for MANETs, we have identified a strong need for middleware services for MANETs to facilitate efficient development of applications over MANETs. Due to space limitations, we could only present a very high-level overview of our work, in which much effort so far has been spent on the requirements analysis. This analysis, which includes also studies of today's approach of rescue teams to collaborate in operations, resulted in the following insights: information sharing is a key element for successful collaboration, knowledge management has to address distribution and different data representation and models that are probably used, information and resources have to be protected with security mechanisms. Furthermore, a decentralized solution is needed that builds on an asynchronous event notification system and uses sufficient redundancy to reach high availability.

We are currently designing five components which provide middleware services to the application and to each other. Our ongoing work is concerned with the particular separation of concerns for the various components and analyzing the tradeoffs in the design alternatives of the components and their particular protocols, like availability versus resource usage. As a next step, quantitative evaluations of various design alternatives for the components will be performed by simulation and emulation.

Acknowledgements: This work has been funded by Norwegian Research Council in the IKT-2010 Program, Project Nr. 152929/431

REFERENCES

1. Alves-Foss, J., An Efficient Secure Authenticated Group Key Exchange Algorithm for Large And Dynamic Groups, Proceedings of the 23rd National Information Systems Security Conference, pages 254-266, October 2000
2. Bresson, E., Chevassut, O., Pointcheval, D., Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case (Extended Abstract), Advances in Cryptology - Proceedings of AsiaCrypt 2001, pages 290-309. LNCS, Vol. 2248, 2001
3. Caporuscio, M., Inverardi, P., Pelliccione, P., Formal analysis of clients mobility in the Siena publish/subscribe middleware, Technical report, Department of Computer Science, University of L'Aquila, October 2002
4. Chen, K., Shah, S.H., Nahrstedt, K., Cross-Layer Design for Data Accessibility in Mobile Ad hoc Networks, Journal of Wireless Personal Communications, Special Issue on Multimedia Network Protocols and Enabling Radio Technologies, Kluwer Academic Publishers, Vol. 21, 2002, pp. 49-75
5. Cugola, G., Di Nitto, T., and Fuggetta, A., The JEDI event-based infrastructure and its application to the development of the OPSS WFMS, IEEE Transactions on Software Engineering, 27(9), 2001
6. Di Pietro, R., Mancini, L., Jajodia, S., Efficient and Secure Keys Management for Wireless Mobile Communications, Proceedings of the second ACM international workshop on Principles of mobile computing, pages 66-73, ACM Press, 2002
7. Fiege, L., Zeidler, A., Gartner, F.C., Handurukande, S.B., Dealing with Uncertainty in Mobile Publish/Subscribe Middleware, Proceedings of 1st International Workshop on Middleware for Pervasive and Ad-hoc Computing, 2003, pp. 60-67
8. Fontijn, W., Boncz, P., AmbientDB: - P2P Data Management Middleware for Ambient Intelligence, accepted for publication in the PERWARE04 Workshop (co-located with PERCOM 2004)
9. Huang Y., Garcia-Molina H., Publish/subscribe in a mobile environment, Proceedings of the 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (mobiDE01), Santa Barbara, CA, May 2001
10. Kärpijoki, V., Security in Ad Hoc Networks, Tik-110.501, Seminar on Network Security, HUT TML 2000
11. Li, B., Wang. K. H., Nonstop: Continuous multimedia streaming in wireless ad hoc networks with node mobility, IEEE Journal on Selected Areas in Communications, December 2003, 21(10), pp. 1627–1641.
12. Meier, R., Cahill, V., STEAM: Event-Based Middleware for Wireless Ad Hoc Networks, Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS'02), 2002, pp. 639-644
13. Montenegro, G., Castelluccia, C., Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses, NDSS'02, February 2002
14. Perich, F., Avancha, S., Chakraborty, D., Joshi, A., Yesha, Y., Profile Driven Data Management for Pervasive Environments, Proceedings 13th International Conference on Database and Expert Systems Applications (DEXA 2002), September 2002
15. Schwotzer, T., Geihs, K., Shark - a System for Management, Synchronization and Exchange of Knowledge in Mobile User Groups, Proceedings 2nd International Conference on Knowledge Management (I-KNOW '02), Graz, Austria, July 2002, pp. 149-156