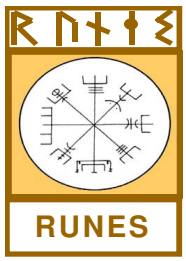


SIXTH FRAMEWORK PROGRAMME
PRIORITY 2
"Information Society Technologies"

Project acronym: **RUNES**
Project full title: Reconfigurable Ubiquitous Networked Embedded Systems
Proposal/Contract no.: IST-004536-RUNES

D4.2
Networking infrastructure and protocols

Project Document Number: RUNES/D4.2/CO[1]/v1.0
Project Document Date: 31/12/2005
Workpackage Contributing to the Project Document: WP4
Deliverable Type and Security: R[2]-PU
Author(s): Miklós Aurél Rónai (editor), Kristóf Fodor (ETH)
Gösta Leijonhufvud, Mattias Johansson, Vlasios Tsiatsis (EAB)
Björn Grönvall (SICS) Gianluca Dini (UniPi), Ida Savino (UniPi)
Wolfgang Fritsche, Karl Mayer (IABG)
Janne Riihijärvi, Frank Oldewurtel (RWTH),
Leonidas Lymberopoulos, Steve Hailes, Manish Lad, Saleem Bhatti (UCL)
Li Li, Louise Lamont, Yasser Gadallah (CRC)

[1] Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

[2]Type: P - Prototype, R - Report, D - Demonstrator, O - Other

Abstract: This document investigates numerous solutions, which will help creating an efficient network architecture within the scope of the RUNES IST project. We hereby introduce a network architecture with special consideration to the road tunnel scenario, which involves monitoring various physical parameters of the tunnel and guiding rescue personnel and passengers in case of emergency. Based on the recent progress on both sensor and ad-hoc network research and development, we can argue that future networked embedded systems will consist of a combination of sensor and traditional wireless ad-hoc networks. This fact leads to network environments that are heterogeneous and at least two-tiered in terms of their communication capabilities. Future networked embedded systems will utilize both low power/capability sensor nodes and high power/capability nodes. For this reason, our architecture covers both sensor nodes, sensor routing nodes and multi-radio sensor routing nodes, including gateways to corporate and public networks. We analyse several solutions for addressing and naming, routing, localization, ad hoc and sensor network interworking, auto-configuration, security and Delay Tolerant Networking (DTN). As far as node addressing is concerned, we first looked at the work that was done by the EU 6th Framework Project Ambient Networks, where an addressing structure is specified in a layered fashion. Some of the existing routing protocols under consideration for our architecture are Directed Diffusion, Geographical and Energy Aware Routing (GEAR), Ad-hoc On Demand Distance Vector (AODV), Low-Energy Adaptive Clustering Hierarchy (LEACH), Two-Tiered Data Dissemination (TTDD), Location based multicast (LBM), GeoGRID and Temporally Ordered Routing Algorithm (GeoTORA). We identified two aspects of the node localization issue: the first one involves localizing the static sensor nodes, while the second involves localizing the mobile agent nodes. We analysed various autoconfiguration approaches; however, further investigation, development, and testing work is necessary to prepare RUNES networks for autoconfiguration. We also discuss network security issues related to the networks and scenarios that we plan to deploy within RUNES. The purpose of our Delay Tolerant Networking work is to implement a message replication protocol that minimizes as possible the number of replicas needed to be created and disseminated in the network; still guaranteeing a satisfactory delivery probability of the messages to their destinations. Finally, this document lists future research directions that we plan to undertake within RUNES.

Keywords: Networked Embedded Systems, Sensor Networks, Routing, Addressing, Localization, Ad Hoc and Sensor Network Integration, Security, Delay Tolerant Networking

History

Version	Date	Description, Author(s), Reviser(s)
1.0	13/01/2006	Released version for submisson

Executive Summary

Our primary goal is to create an efficient networking architecture for distributed networked embedded systems using the fire in the road tunnel as an example scenario. In this document, we introduce the architecture and investigate several solutions for various aspects of networking. In the following, we will briefly outline the issues addressed in this document.

Architecture

In our architecture we introduced *sensor nodes*, *sensor routing nodes* and *multi-radio sensor routing nodes (or gateways)*. The sensor and sensor routing nodes are equipped with sensors, which are measure various parameters of the environment. Basically we assume that the tunnel's infrastructure is wired, but all devices have fallback radios for the case of an emergency, when, e.g., fire breaks out, and destroys wires and nodes. The sensor nodes are deployed in such a way so that initially no network partitions exist. That means that every node can potentially reach every other node by a wireless multihop path. Sensor routing nodes can reach each other and some of them can communicate with multi-radio sensor routing nodes, as well. Multi-radio sensor routing nodes have at least two radio interfaces, one to communicate with sensor routing nodes and another one to communicate directly with each other. Some of these multi-radio nodes have external connections towards the Internet.

Addressing

To address nodes in this architecture first we looked at the work that was done by Ambient Networks, where an addressing structure is specified in a layered fashion. This solution can be seen as a general way to structure the addressing issue, to help simplify the understanding of different addressing schemes and to more clearly differentiate between, e.g., "routing addresses", "abstract naming", "geographical addressing", "functional addressing", "application addressing", etc. It is not suggested that we shall follow the Ambient Network structure literally but it is obvious that we need a way to define not only basic connectivity but also the structure of distributed applications in RUNES. In a sensor node we need to collect sensor data and we might need to filter these sensor data. Higher up in the application hierarchy we may use additional filtering, fusion, intermediate storage of data etc. This application structure represents an application "infrastructure" that require its own "application routing".

Routing

We investigated some of the existing routing protocols, which could be used in our architecture, such as Directed Diffusion, Gear, AODV, LEACH, Two-Tier Data Dissemination, Location Based Multicast, GeoGRID and GeoTORA. When we consider the conditions that should exist in the routing protocols for sensor networks, we find that none of the overall characteristics of the investigated protocols present a perfect match for the conditions of sensor network operation. In the document we discuss the pros and cons of using these protocols. We decided to have a closer look at them in the future and to think about enhancements to make them suitable for our purposes.

Autoconfiguration

In the PC and server world much of the configuration work today is still be done by humans, either administrators or the users themselves, however, sensor / actuator nodes are embedded devices with limited configuration possibilities via human machine interfaces (HMIs). Furthermore the majority of their users would not have the expertise to perform the required configuration. Consequently one needs to investigate how the required configuration tasks as described above could be handled. In RUNES networks autoconfiguration of network nodes is highly desirable since nodes may provide no or only limited human machine interfaces, networks may scale to thousands of nodes, and RUNES

networks are likely to be deployed in an ad hoc manner with no pre-existing infrastructure at all. In the document we investigate several approaches for autoconfiguration.

Localization

In the RUNES project, two types of localization problems are identified. The first problem is to locate the static sensor nodes and the second issue is to locate the mobile agent nodes. The pre-installed sensor nodes in the tunnel or building may have known location-ID mappings that can be loaded on the nodes at program or run-time. Thus upon its initialization, the node can come up with its location information. These are the anchor nodes. There are other randomly deployed ad hoc sensor nodes, for example, the special sensors deployed by the fire first responders when entering the tunnel that is on fire. These sensors need to locate themselves in time to report readings associated with their location and to assist in tracking the mobile agents, including robots, fire fighters and rescue vehicles. The approach under investigation is to apply RF signal sensing to establish the location information of the ad hoc sensor nodes. Though RF sensing may not be accurate in an indoor or tunnel environment, pre-planning can often establish the signal patterns for location references.

Ad hoc and sensor integration

Based on the recent progress of sensor network research and development we can argue that future sensor networks will be heterogeneous and at least two-tiered. Sensor networks will consist of basic sensor nodes and nodes with significantly higher communication capacity and energy sources than the basic sensor nodes. Such networks will have two advantages compared to flat networks. First, overall capacity of the network will be improved by the more powerful radios of some special, “enhanced” nodes. Second, the network will be hierarchical, thus it will be easier to manage. This will be one direction of our future research. Another major research issue is horizontalisation. We believe that in order for sensor networks to become pervasive, it is important to find and exploit horizontalisation points on which development can grow. At the same time, one must not forget that in order to save energy, control over the underlying technologies is important. We believe that it is important to allow multiple applications utilizing different network protocols to exist in the same sensor network. Different administrators might also want to use the same sensor network infrastructure using different address spaces for their networking protocols. In order to achieve this possibility, we propose to let a link layer abstraction take care of the multi-hop routing. The network address is only processed on those nodes that are running the particular application that is utilizing this networking protocol. Another possibility is that the node should belong to the particular administrator using the specific address space. This implies that some kind of routing has to be performed on the link layer abstraction plane. These topics will be further investigated in the future.

Meshed topologies

The meshed topologies being formed in RUNES are essentially multi-homed with several ingress and egress connections to the wide-area (i.e., Multi-radio sensor routing nodes that have 2G/3G connectivity). Administrative responsibility may be distributed across the nodes forming the network. For example, the sensors that line the tunnel wall may be owned and controlled by local government authorities, but the sensors that ride on vehicles are owned and controlled by the vehicle owners themselves. Thus, although the different groups of devices may interconnect, they do not necessarily represent a single Administrative Domain (AD) that is under the control of a single organisation or entity, but rather a collaborative group of such entities. However, existing mechanisms for addressing and routing are designed to operate in network environments where administrative responsibility is not distributed, and therefore may not provide the most optimal solution for interconnectivity.

Security

The architecture possibilities for sensor networks that we consider in RUNES have two dimensions – the location and duties of policy enforcement points, and the choice of WAN connection method.

Enforcement has to be implemented even in the smallest sensor nodes - but this enforcement can be very simple and based on proving knowledge of a secret key. For instance, one or more powerful nodes might control a simple sensor node. The more powerful nodes then need to make the decision and enforce which applications and users that have access to the less powerful nodes. Beside this access control will be investigated in the future.

Mobility of RUNES gateways

In case the gateway nodes – for example the nodes that interconnect the vehicle's infrastructure with the outside world – of RUNES sensor networks are IP enabled and their mobility is handled on IP level, then the IETF has specified various solutions in the area of host and network mobility. In this document we introduce these solutions and assess how they can support various mobile sensor network scenarios.

Delay Tolerant Networking

There is a growing interest in the Internet community in Delay Tolerant Networking communication mechanisms as a means of implementing message delivery in extreme and performance-challenged network environments, where end-to-end connectivity cannot be assumed. In such environments, one should be still able to guarantee delivery of messages to their destination in order to support operation of his/hers applications. Examples of such environments include spacecraft, military/tactical, some forms of disaster response, underwater, and some forms of ad-hoc sensor/actuator networks. In a resource-constrained environment, such as a RUNES sensor network, where sensors have finite and small-size buffers, there is the need to reduce the number of replicas in the network as much as possible. The purpose of our work is to address this issue by designing and implementing a protocol that takes into account the "{m,n} hop" metric, which is used to specify how close a message is to its source (variable m) and at the same time, how close the message is to its destination (variable n).

- **Conclusion**

Concluding, this document covers lot of research topics and contains many possible research directions we may conduct in the future. Our aim is to provide a set of networking solutions cooperating with the RUNES middleware (developed by WP5) that will support communication between heterogeneous embedded sensor devices. Our goal is to provide a as much as possible generic RUNES solution, where several networking approaches will live together to best fulfil the requirements of various applications and networking environments.

Contents

Figures, Tables

1 Introduction

In the previous deliverable [20], we investigated the networking requirements for RUNES. In this document we investigate various networking approaches that could be applied for RUNES sensor networks and fulfil the requirement of the "fire in the road tunnel" scenario [1]. The solution for the fire in the road tunnel scenario should include monitoring of road tunnels and supporting functions for rescue services in case a disaster happens.

In our future scene, wireless technology can assist in the control of goods in the tunnel – all cargoes are required to carry an RFID tag with information on the contents, amount of material and hazard information. This means that cargoes which have the risk of violent explosion or other associated hazards can be refused passage or allowed through under restrictions, since load details can be accessed remotely.

We assume that air quality monitoring equipment has been installed within the tunnel. This equipment measures temperature, humidity and the concentration of certain gases. Traffic flow can be restricted within the tunnel to ensure that air quality remains acceptable; however this can result in traffic problems around the entrance and exit. Information on air quality is made available to anyone who may be concerned about the potential hazard to their health (e.g., asthmatics) via a suitably-equipped PDA.

Consider a busy weekday with significant traffic in the road tunnel. Let us assume that a collision happens deep within the tunnel between several vehicles including a tanker loaded with heated vegetable oil. The tanker suffers penetration of the tank and begins to leak, spreading oil all over the road surface. A small fire started as a result of the collision spreads to the oil, which begins to burn producing clouds of thick smoke as well as heat and flame. The detection system within the tunnel picks up the fire and immediately triggers an alarm. The tunnel is closed for incoming traffic and the emergency services are summoned. Management of the resulting traffic congestion will be started.

Virtually all the tunnel users have personal communication devices. Some people call the emergency services for help, some seek information from the local information systems located at the tunnel emergency refuges. Later on, the emergency services may contact some of these people within the tunnel to obtain information on or even pictures of the scene of the accident.

Some of the tunnel users have medical problems which result in them carrying personal medical monitoring equipment. Normally, a person with a medical problem would prefer to keep this information confidential, however the nature of the emergency means that some of those involved will require priority evacuation or the trauma of the incident might have additional consequences because of their conditions.

The first challenge to the emergency services will be access to the tunnel. The traffic resulting from closure of the tunnel will present a serious delay to any road vehicle. Future infrastructures may allow traffic signals to be changed to facilitate access or information from vehicle networks to identify the fastest possible route. Commuters involved in the resulting traffic jams but not affected by the fire risk will also require information on the cause of the problems and anticipated delays or alternative routes.

On arrival, the immediate priorities will be to establish the situation inside the tunnel, to rescue the people inside the tunnel and to tend to the injured. To understand what the situation is within the tunnel, the rescuers will need to be able to use existing sensors available in ways, which their current software may not support. For example, the temperature and air quality sensors may take readings, which would normally be considered erroneous due to being too high. There may be a need to download data from vehicle on-board systems, which would not normally make data available. The

emergency services therefore need to be able to download software, which will permit use of the sensors available to them in whatever way they require.

Once the situation is sufficiently understood the first rescue workers will enter the tunnel. Where the risk is high such personnel will have personal health monitoring equipment to allow the command and control outside the scene to recognise risk to their operatives. An alternative possibility is the use of mechanical devices, which can be sent into the tunnel to locate hazards such as fire or leakage of dangerous cargoes. Their sensing systems could be added to the network as required.

Some rescue workers may have sensing devices not carried by others providing data on, for example, particular noxious chemicals or readers for the RFID tags on cargoes. Rescue workers may also require temporary medical monitoring information on victims. The data generated must be made available to all workers. Medical workers will be required to deal with victims of the tunnel disaster itself and also any injuries to the rescue workers.

The problems associated with different emergency services having communications systems that do not allow them to talk to one another over their radios are already familiar. These problems must be addressed in the context of personnel who have not only radios but also personal telemonitoring, handheld sensor devices and in the case of paramedics, telemonitoring equipment for use with casualties.

In traditional distributed security, authentication is established after contacting a trusted authority responsible for maintaining up-to-date record of each user's access rights. However, in a disaster response scenario, communication with this authority might be poor or even impossible. In such a case a best-effort security model may be more appropriate. Approaches based on public key cryptography overcome the need for a trusted authority but the computational requirements are high considering the resources available on sensor nodes. Algorithms implementing elliptic curve cryptography are believed to be more computationally efficient than the former.

In order for the surviving parts of the sensor and communication network to remain in a healthy state for enough time to allow the rescue mission to be completed successfully the nodes must be supplied by power. Replacement of batteries is often impossible or impractical and hence any solutions relying on power scavenging would be of great value. Finally, constraints in wireless communication include bandwidth shortage particularly when the unlicensed 2.4GHz frequency band is considered.

The rest of the document is structured as follows. First in Section 2 we introduce the sensor networking architecture for the fire in the road tunnel scenario. In Section 3 we discuss what kind of addressing and naming scheme could be applied in RUNES sensor networks. Section 4 analyses various sensor routing mechanisms and Section 6 is about autoconfiguration. Section 7 investigate routing challenges in meshed topologies and Section 7 elaborate various localisation techniques. Section 8 discuss how sensor networks can be integrated with ad hoc networks and outlines research challenges related to this topic. Security and access control problems are analysed in Section 9. In Section 10 the mobility issues of RUNES gateways are discussed and in Section 11 a new delay tolerant networking message replication approach is introduced. Finally in Section 12 we conclude our work.

2 Scenario and network topology

To improve human security *sensor nodes* (small blue circles), *sensor routing nodes* (yellow circles) and *multi-radio sensor routing nodes* (red rectangles) are deployed along road tunnels (see Figure 1). The sensor and sensor routing nodes have sensors on them, which are measuring various parameters of the environment in the tunnel, such as air quality, temperature, humidity; some of them can detect fire. Basically this is a wired infrastructure, where all devices have radios for fallback for the case of an emergency, when, e.g., fire breaks out, and destroys wires and nodes. The sensor nodes are placed to reach neighbouring sensor routing nodes in one hop distance. Sensor routing nodes can reach each other and some of them can communicate with multi-radio sensor routing nodes, as well.

Multi-radio sensor routing nodes have at least two radio interfaces, one for communicate with sensor routing nodes and another one to communicate directly with each other. Some of these nodes have 2G/3G radios (red rectangles with white rectangles inside), through that they can connect to the internet, thus playing the role of a gateway and providing internet connection to the tunnel's infrastructure. Multi radio sensor routing nodes act also as a collector for sensor data. In the picture the big circles represent the radio coverage of the different devices.

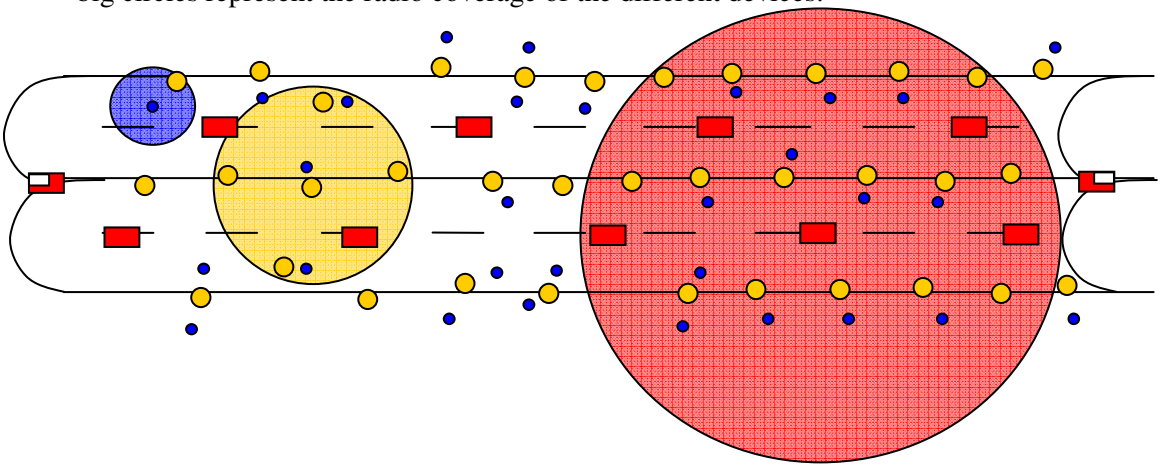


Figure 1: Sensor, sensor routing and multi-radio sensor routing nodes in the tunnel

In case fire destroys some nodes and cables (see Figure 2) the concerned nodes switch to radio communication and establish a new routing topology. The black arrows show, that the concerned sensors, which lost their collector nodes, had to search for an other multi radio sensor routing node and had to change the destination of their packets. The black lightning sign shows, that a radio link is established to keep up the connection between the two parts of the tunnel's network.

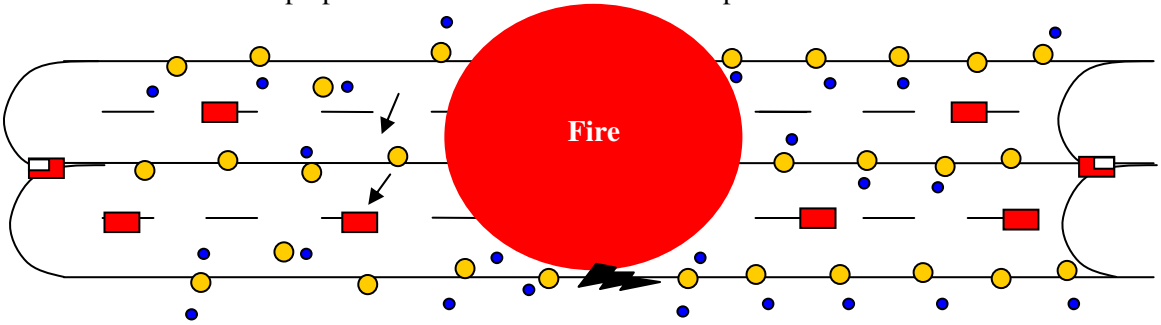


Figure 2: Fire in the tunnel destroys some parts of the network

It may happen that most of the tunnel's network is destroyed. In this case firemen can deploy new nodes if necessary and possible. The newly deployed devices (green circles) can connect via radio to

the devices which survived in the tunnel (see Figure 3). A strict access control mechanisms ensures that only the sensor nodes of the firemen are able to attach to the tunnels infrastructure this way.

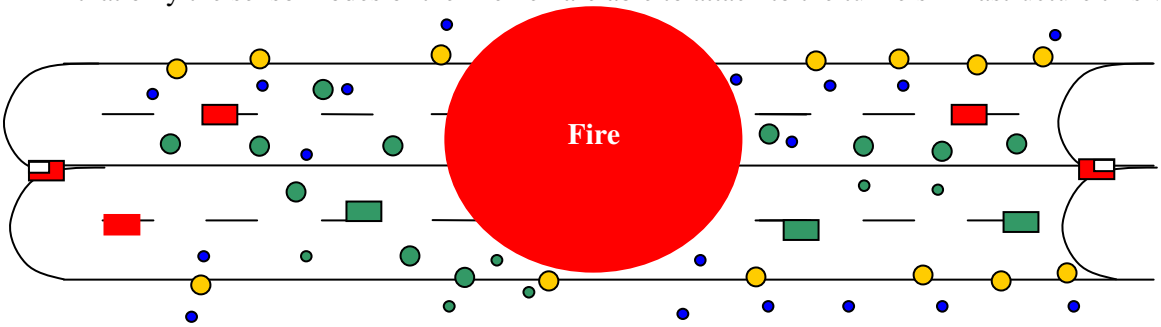


Figure 3: Fire in the tunnel destroys most of the network

Vehicles (cars, trucks, fire-engines, etc.) also have sensor nodes, a sensor routing nodes and multi-radio sensor routing nodes (see Figure 4). If vehicles are in the tunnel, they may connect to the tunnel's infrastructure through their gateways in case of emergency. This way the firemen will be able to query the sensors in the vehicle to gain information about the situation around the vehicle. Passengers in the vehicle may download the tunnel's map to learn where the fire exits and fire safe rooms are located, and which of them are still available.

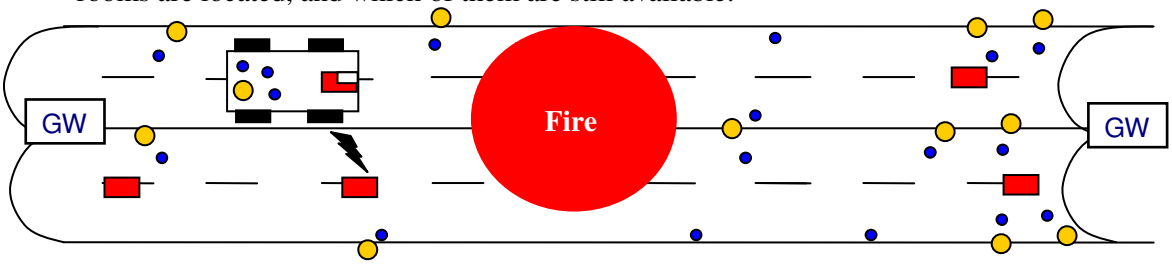


Figure 4: A vehicle connects to the tunnel's infrastructure

2.1 Network topology

To summarize the scenario above the following network topology can be depicted (Figure 5). The lines between the nodes show the radio connectivity type, so the dotted red lines mean a different radio technology (e.g., 802.11) than the black lines (e.g., 802.15.4). We assume that basically all nodes are wired together with Ethernet, and are powered by Power-over-Ethernet.

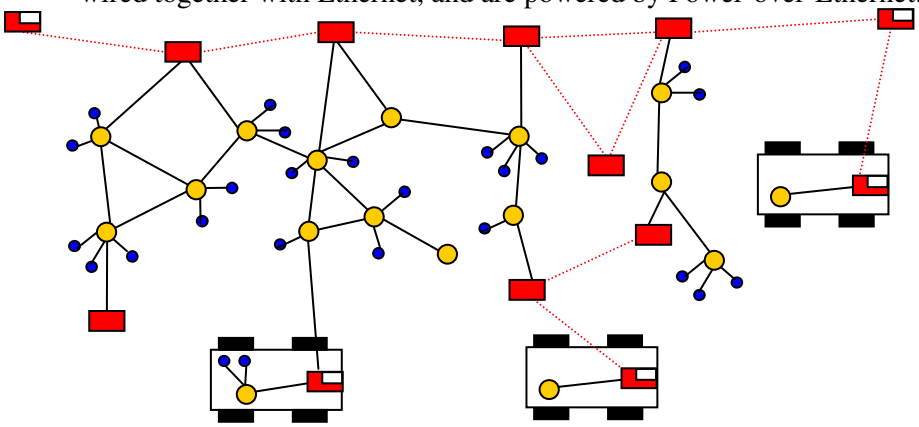


Figure 5: View of the topology by radio connections

3 Addressing and naming

Within the IST project "Ambient Networks" [2] they have specified an addressing structure in a layered fashion. It is brought up here since it can be seen as general way to structure the addressing issue and will hopefully simplify the understanding of different addressing schemes and to more clearly differentiate between, e.g., "routing addresses", "abstract naming", "geographical addressing", "functional addressing", "application addressing" etc.

The following figure below suggests four layers for different purposes. Observe that we might consider splitting layers in sub layers. Examples could be that we model the connectivity layer in a layer 2 routing layer using link layer addresses and a layer 3 routing layer using IP. We may also consider a hierarchical structure of a distributed application. The text and figures below is copied from AN-D1-5 "AN Framework Architecture".

D1.5: s.2.4 Connectivity Abstractions and Ambient Naming Framework

The connectivity abstractions defined above map to the Ambient naming framework as follows:

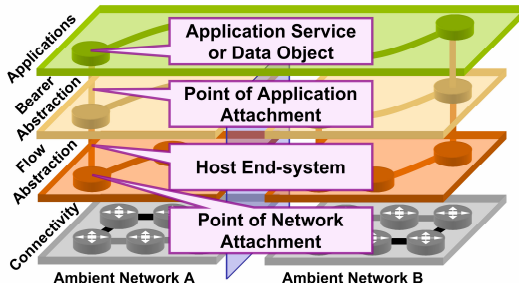


Figure 6: The Ambient Networks connectivity abstractions and their relations to the naming objects

- **Application service or data object:** An entity that is either a specific application service, or a specific data object. The identity of the object persists over time and is not tied to the end-system hosting the service/data. Examples of names in this layer include are HTTP and SIP URLs. These names are used by applications as references which are resolved and result to creation of application sessions utilizing the lower level connectivity abstractions.
- **Point of application attachment:** The point, where clients can reach an application program that implements parts of an application service. This point is located at the ASI and can be compared to a standard TCP/IP socket API. A bearer endpoint is instantiated at a point of application attachment when a bearer setup is completed.
- **Host end-system:** Network topology independent identification of a node in an Ambient Network, which does not necessarily mean a physical box – it may be a logical entity that can move between physical boxes. The host end-system is the entity "hosting" the ASI. Cryptographically generated identifiers present an example of secure, self-assigned host end-system identities. The host end-system abstraction makes address/protocol translations on the locator level invisible to the point of application attachment.
- **Point of network attachment** – Defines a location in the network. The location is identified with a locator, which is a routable address in the underlying network technology, such as an IP addresses. This point is exposed to the ACS via the ARI. A flow, the lower level AN connectivity abstraction, runs between two network points of

attachment.

Each flow is restricted to a single locator domain and bridging or translation at the locator domain boundary is needed to support communication between any nodes in the Ambient Network. This translation is best managed via mapping of host end-system identities to locators in each locator domain passed.

The naming framework supports the notion of *indirection* or *delegation*. It is an extension to the dynamic bindings that enables more advanced mobility schemes and the explicit control of so-called middleboxes, "Address and protocol translation gateways", such as network address translators firewalls and transcoders.

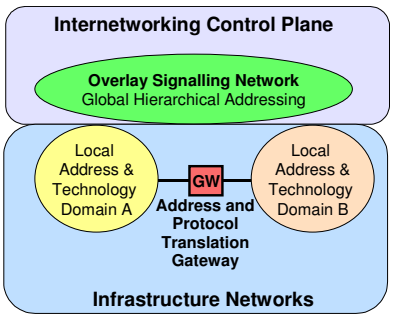


Figure 7: Internetworking principle of the overlay internetworking architecture. The internetworking control plane controls the gateway.

3.1 Address and naming structure for RUNES

Observe that it is not suggested that we shall follow the Ambient Network structure literally but it is obvious that we need the capability to use proprietary addressing schemes (locators) within a sensor domain for forwarding and routing and still be able address individual sensor by global naming.

Ways to define not only basic connectivity but also the structure of distributed applications in RUNES are also required. In a sensor node we need to collect sensor data and we might do filtering of the sensor data. Higher up in the application hierarchy we may use additional filtering, fusion, intermediate storage of data etc. This application structure represents an application "infrastructure" that require its own "application routing".

Looking at the general sensor configuration for RUNES we have the following structure copied from D1-3. Three types of nodes are defined within the sensor domain namely, gateways, sensor routing nodes and sensor nodes. The sensor and sensor routing nodes will form the "Sensor/Actuator Network" in the figure below. The Gateway will represent the bridge between the "Wide Area Network" (public or corporate) and the "Sensor/Actuator Network".

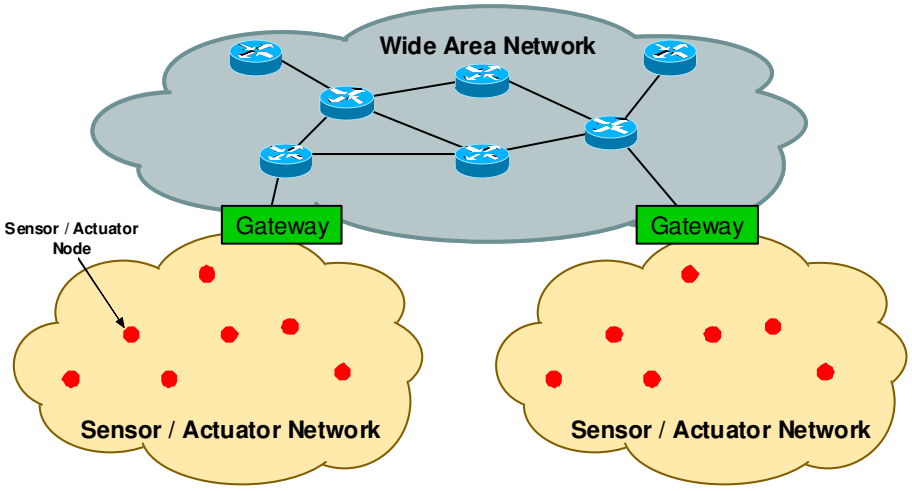


Figure 8: Connecting two sensor networks

From an addressing point of view we are able to separate the Wide Area Network from the Sensor/Actuator Network the different Sensor/Actuator Network can be separated from each other at least if they are supporting different applications or service areas. (This does not exclude the option to use a public addressing scheme within a specific Sensor/Actuator Network).

A gateway has to have a public address "Point of network attachment" while the individual sensors including the gateway may use a private addressing (Point of network attachment) scheme for communication within the sensor domain.

For the public domain we assume that we are bound by existing addressing schemes while we internally within the sensor domain can use private addressing schemes, e.g., specially designed for the sensor network requirements of RUNES.

A sensor node that can move from a sensor domain and require the capability to access a sensor application through a public network do also require a public address and has to be able to act as a gateway.

3.1.1 Abstract addressing

From a connectivity point of view the sensor domain will typically be fully meshed, but from an application point of view it will typically be hierarchical. In the latter case the individual sensor (the data source) will communicate with the "control logic" and the "control logic" may communicate with the actuator or with secondary "control logic" higher up in the application hierarchy. An example is given in the figure below.

Applications that can be structured in a hierarchical way simplify the network and improve the scalability.

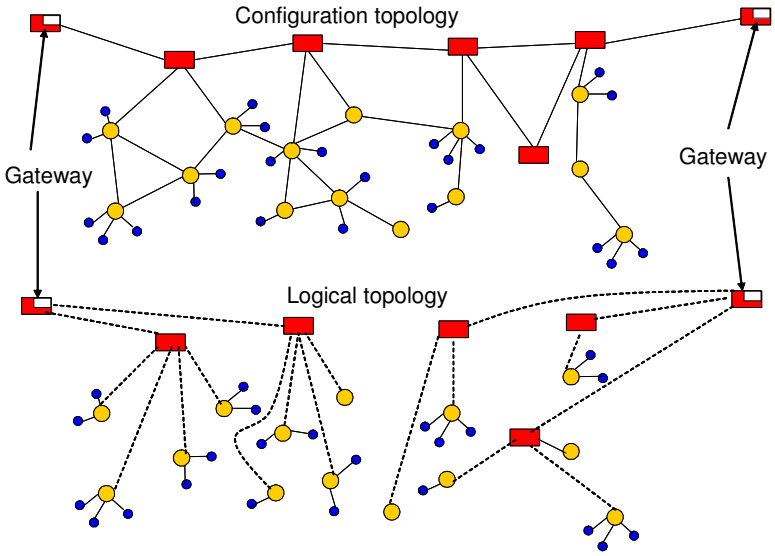


Figure 9: A configuration topology and logical topology of the same network.

Here it is assumed that we have relevant control applications running in the rectangular sensor routing nodes as well as in the gateways. All sensor nodes are communicating to the nearest control application which is secondarily connected to two gateways.

From an addressing point of view it will be required that we have a name or address for those application (functions) that can be mapped to the actual physical node. Observe that all application nodes might house multiple applications and that the logical communication topology may be different for different application

Within the sensor domain we are able to define our own abstract naming. We will require addresses not only for applications and functions but also for geographical addressing and we will need functionality to resolve such abstract names.

3.1.2 Other Addressing and Naming issues in RUNES

Generally all entities within the sensor domain (Sensor Node, Sensor Routing Node or Gateway) need the capability to be globally identified. As global identifier we might use MAC address or DNS names as globally resolvable names that can be used for authentication.

In many cases IP addresses is considered to be globally significant. Still IP addresses can not generally be used to identify a host or node since they may be a given a new IP address for instance when moving to a new location.

Routing addresses are addresses that are used by the routing program and they only have to have be significant within the routing domain. For sensor domains like within RUNES we may have short fix or variable length addresses with limitations to a preconfigured domain. IP addresses as extension to the public domain may also be used but is not required to support a close user domain.

A routing address is an address that can identify a port within the sensor domain and can be auto configured in start up time and in case of reconfiguration.

A sensor domain will typically support specific applications (one or more). Functional addressing will therefore have a specific meaning in each domain. Examples are functional addressing requesting

temperature or humidity readings in a specific geographical area. The structure of such functional and geographical addressing may have to be application or sensor network specific.

Sensor applications will be distributed hierarchically and we need an abstract naming structure for the application. The addressing may match the hierarchical structure e.g. <application> <sensor domain> <function>. This is to be defined at the time for configuration. You will also need a way to address the server or gateway from a public / corporate terminal.

As the gateway represents the interface between the public (or corporate) domain and the sensor domain it has to match the public address space as well as the sensor address space. It will also have to support NAT functionality when needed.

3.2 Available addressing mechanisms

3.2.1 Naming of entities

Naming systems have been used long in the Internet as a complement to the IP addresses. As the scale of the Internet extends, DNS (Domain Name Service) is used to organise hosts into domains and to resolve host names to IP addresses. Although the IP addresses can indicate network topology, a domain name seems more intuitive for people to remember. Sensor networks may have large scales, so a naming scheme can also help organising the sensor nodes.

A lot of work has been done in attribute-based naming. In the Internet, attribute-based naming system such as X.500 [63] and LDAP [64] have been used, and provide hierarchical architecture. The Intentional Naming System (INS) [65] uses an attribute-based name to describe different types of services. Intentional Name Resolvers (INRs) are used to resolve IP addresses of the services or forward requests to the service nodes. These protocols are based on the IP layer, and require directory services for resolving names and addresses. Directory service may introduce more communication overhead, so these naming schemes cannot be deployed in the sensor network directly.

In [66] the authors proposed a low-level naming scheme based on attributes that are external to the network topology and can be specified in applications. In this scheme, the named data can be transmitted hop-by-hop toward the destination without relying on IP addresses. The named data packets are self-identifying, which can enable in-network processing for data aggregation, and thus reduce data transmission in the sensor network. The architecture of the low-level naming contains directed diffusion, matching rules, and filters. Directed diffusion is used for data propagation in the network. Data is managed as a list of attribute-value-operation tuples. Matching rules determine if the data has arrived at its destination, or it should be processed in the intermediate nodes. Filters enable application-specific code to run in the network and assist diffusion and in-network processing.

3.2.2 Types of address auto-configuration protocols

Address auto-configuration protocols in general can be categorized based on the manner they treat the state of the address space. Those approaches where none of the participants in the network maintain information about the reserved addresses are the *stateless* solutions, while the ones which make an effort to store a quasi up-to-date version of the address space are called *stateful* solutions.

Existing addressing schemes for auto configuration of ad hoc networks in general are further described in a separate chapter below (Section 6.1). Here it is assumed that all nodes shall be able to communicate with any other node and those concepts are limited to connectivity and to the addressing required for the basic routing forwarding level. The fact that sensor application are hierarchical by

nature is not considered in those schemes. With an overlay concept as indicated above we will be able to separate “application routing” from the basic routing.

3.3 Conclusion

From a RUNES perspective we need to evaluate existing addressing schemes against the requirements. If we find out that specific issues is not covered we might consider to upgrade one of the addressing schemes or design a new.

We also need to define how to support functional and geographical addressing within a RUNES sensor domain. Those dedicated addressing schemes will be part of an overlay structure.

Addressing is also effected by the security mechanisms that might be required from a privacy and integrity point of view. For example, special authentication mechanisms may be required. This is further addressed in Section 9.

4 Routing

Routing in sensor networks has the main function of ensuring that the information gets transferred successfully from network nodes that "sense" this information from their targets to the end destination where it gets analyzed and utilized. To pave the way for this discussion, we make the following main assumptions about the network as well as network nodes' characteristics:

- The network may or may not be homogenous in the sense that its nodes may not necessarily have the same computational and storage capabilities as each other. This means that some nodes may have stronger computational capabilities than others within the same network
- Sensor nodes have limited resources. This includes:
  - Energy resources (batteries),
  - Memory,
  - Storage, and,
  - Processing capabilities
- Wireless communication range for some of the sensor nodes is limited
- Network topology is expected to change with time, in general. This could be due to failed nodes or new nodes joining the network or due to sink node mobility
- Number of nodes within the network is generally large (from few hundreds to few thousands)
- In general, several types of target data are required to be collected by different sensors within the same network. The end point for sending this data could be one or more 'sink' (or destination) nodes

Based on these assumptions, we can extract some conditions that should exist in the routing protocol that is to be selected for this type of networks:

- The overhead of the routing protocol specific traffic should not be significant to avoid overloading the bandwidth constrained network or affecting network throughput. It should also not impose heavy computational, storage or memory requirements on network nodes
- The routing protocol should be able to scale with the size of the network population and traffic
- The routing protocol should either be inherently energy-efficient, or is able to integrate with some algorithm that enables it to achieve this goal. Energy-efficient operation implies using node energies both wisely and fairly and avoiding dissipating this energy unnecessarily
- The functionality of the routing protocol should be as distributed amongst as many nodes as possible, for robustness. Therefore, the protocol should not be dependent on a node or a small subset of network nodes for its functionality
- The routing protocol should be able to respond to changing network conditions and topology promptly and efficiently without imposing heavy traffic load on the network as a result

As we can see, most of these conditions stem from the nature of sensor networks and the limited resources that are available to their nodes. In a sense, we ideally want to use a routing protocol that does the job of connecting the source and destination nodes with almost no cost. In reality, it is not possible to find a routing protocol that achieves all these required aspects with no side effects or costs.

The goal then is to find or develop the protocol that has the features that fit these conditions best while imposes the least possible cost or side effects on the network environment and operation.

In this chapter, we discuss both unicast routing as well as geocast routing, which is defined as geography-based multicast routing, in sensor networks. We also give some examples of protocols that belong to the different groups and discuss their suitability based on the above discussed criteria.

4.1 Routing in sensor networks

Many techniques have been created specifically for data transfer within sensor networks. These techniques can be broadly classified into the following two categories:

- Flat-structured protocols
- Hierarchical or cluster-based protocols

It is worth mentioning that routing protocols that belong to same category may resort to using different techniques, e.g. some techniques may be based on geographical locations of network nodes while others may function independently of node locations, etc. Conversely, we may find that the same technique is utilized by routing protocols that belong to different categories. The examples that we will use with each category will highlight this observation.

In the following sections we describe each of the above categories together with example algorithms. We have selected the protocols that can potentially accommodate the tunnel scenario described earlier in this document. Further more, our subsequent evaluation work of these protocols will consider the elements of this scenario, such as different node capabilities and mobile gateways or sink nodes.

4.1.1 Flat-structured protocols

In this category, none of the network nodes or subset of nodes are expected to carry out duties on behalf of other network nodes. Therefore, all nodes are considered to have the same weight or rank at any point of time with regard to the functionality of the algorithm. The differentiator between nodes would only be the suitability of the node's position or state for a certain route or path.

As we discussed earlier, protocols that belong to this category use different routing methodologies. For example, some of them are independent of node locations (geography) while others rely on geographical locations for their functionality. In the following, we describe two of the algorithms that belong to this category but use different techniques: the directed diffusion algorithm and the GEAR algorithm.

4.1.1.1 Directed diffusion

In the directed diffusion algorithm, communication between nodes is done based on a per-hop basis. Therefore, no elaborate routing is done. Sink nodes decide on which data is needed and the intervals at which this data needs to be transmitted. Different semantics can be given to gradient values depending on the task that the operator wants to perform. Basically, the concept of the gradient, which mainly describes the data, its dissipation frequency and its flow direction, is used to handle data transfers. The process starts by a sink node broadcasting its need for certain data that is described by specific attributes. This request gets propagated through the network until it reaches a node that either possesses this data or knows where to get it. When several responses reach the requesting node, a neighbour is reinforced or selected based on data driven local rules. The most common rule is to select the neighbour from which the fastest response was obtained. The original interest message is resent through the network at a higher data rate to select a certain data path from multiple paths.

4.1.1.2 Gear

Another example of this category is the GEAR algorithm. It stands for "Geographical and Energy Aware Routing". It works on ensuring that routing is directed only towards geographical regions of interest. It is assumed that each node knows its own location and remaining energy level as well as those of its neighbours. A node forwards a packet based on the proximity of the target region and also on the energy balance for the neighbours. This is done based on a learned cost value of the neighbours of the node. If this learned cost is unknown, an estimate is used which is calculated based on the distance from the target and the energy consumed at the neighbour for which this estimate is being calculated. When there exist neighbours that are closer to the destination than the current node, minimizing the learned cost becomes a trade-off between routing towards the next hop closest to the destination and balancing energy usage. When all neighbours are farther away from the target than the current node, it is considered to be in a 'hole'. The node tries to circumvent this hole using its knowledge of its cost as well as its update rule, which it uses to update its knowledge about the cost to the required target, to route the packet. When the packet reaches the required region, a recursive geographic forwarding technique is used where the region is subdivided into 4 sub-regions and the same technique as described above is repeated, and so on. The propagation stops when a node finds itself alone into a sub-region. This occurs when its neighbours that are within its wireless range are all outside the region.

4.1.1.3 AODV

The functionality of the AODV protocol is based on maintaining a vector of paths (i.e. routing table) that lead to the different destinations at each node. A given node does not have a full knowledge of any of the routes. It only knows the next hop along any given route. Each node keeps only one route (the one that has the smallest number of hops i.e. shortest route) to any given destination. When a node needs to communicate with another node to which it does not have a route, it sends a route request to its neighbours. This request gets propagated in the network until a node that knows a route is found. It then replies back with the info about the requested route. The classic AODV algorithm also uses periodic HELLO messages to keep neighbours aware of the other nodes in the neighbourhood. In more recent versions of AODV, this is done via relying on link state capabilities of the underlying MAC protocol. This procedure is used for route maintenance. The AODV protocol does not have inherent energy-conservation capabilities.

4.1.2 Hierarchical or Cluster-based protocols

Routing protocols that belong to this category usually organize the network into clusters or cells of network nodes, with each cluster having a specific node or nodes designated to act on behalf of the cluster nodes. These nodes are sometimes called the cluster heads. The designated nodes within a cluster are in general responsible for communicating the data of their cluster nodes or exclusively handling data forwarding to the destination or next hop in the structure. There are several possible criteria for selecting the cluster head. For example, it could be selected based on its remaining energy. Alternatively, the proximity to the sink node can be used as the selection criteria, and so on. Usually, the main goal of this category of algorithms is to achieve energy efficiency by decreasing the number of nodes that have to contribute to data transmissions to the sink. In the following, we describe two of the algorithms that belong to this category but use different techniques to carry on the routing duties.

4.1.2.1 Leach

One of the main algorithms that belong to this category is the LEACH (Low-Energy Adaptive Clustering Hierarchy) algorithm. In this algorithm, network nodes organize themselves into clusters. Each cluster has a cluster head. The cluster head, which will handle communications with the sink node on behalf of cluster nodes, gets selected based on a certain scheme that depends on determining a certain percentage of network nodes that would be acting as cluster heads at a given point of time. Nodes that have not become cluster heads in the previous rounds are the candidates to become cluster heads in the upcoming rounds until all nodes are covered. After cluster heads have been selected, they

advertise their new status and nodes decide to which cluster they would belong based on the signal strength of the advertisement packets that reach them from the different cluster heads. In case of a tie, the cluster selection is done randomly by the node. Once clusters have been established, each cluster head establishes a TDMA schedule for its cluster nodes for sending their data to it and inform the nodes when they should transmit by broadcasting the schedule within their own cluster. The cluster heads perform local data fusion and send the information to the sinks (gateways). When it is the time slot for a certain node, it sends its data to its cluster head. Otherwise, it turns its radio interfaces off thus saving its energy. LEACH also uses CDMA coding to avoid communication interferences between adjacent clusters.

4.1.2.2 TTDD

Another technique that belongs to this category is the TTDD (Two-Tier Data Dissemination) algorithm. The TTDD technique uses a grid structure in such a way that only sensors located at grid points need to acquire the forwarding information. The TTDD design is built on the assumption that network nodes are stationary and location-aware. A source node detects an event and proactively builds a grid structure throughout the sensor field and sets up the forwarding information at the sensors closest to grid points (dissemination nodes). When a sink issues a query, it traverses the grid structure through the dissemination points towards the source of the information. Each data source sets up its grid structure that is used for its data dissemination. Therefore, sinks will need to flood their queries only within their local grids or cells until a dissemination node is reached. The propagation of the query will be done through the dissemination nodes for the requested data. Once the query reaches a local dissemination point, it is forwarded on the grid upstream towards the source. Each dissemination node along the path keeps information about the downstream node from which it got the query. Data is then sent back from the source to the requesting sink along the same path. TTDD handles sink mobility via assigning changeable *primary* and *immediate* agents to each sink node. The primary agent takes care of communicating with dissemination nodes. It then sends the data to the immediate agent which in turn relays it to the sink. When a sink node starts moving out of range of its immediate agent, it acquires a new immediate agent and informs its primary agent. Similarly, the primary agent is replaced when the sink moves away from it.

4.1.3 Suitability of the different routing techniques for sensor networks

When we consider the conditions that should exist in the routing protocols for sensor networks as we outlined above together with our discussions about the different protocols, we find that none of the overall characteristics of these protocols present a perfect match for the conditions of sensor network operation. This can be clarified by the following discussion of protocol shortcomings.

Considering the Directed Diffusion algorithm, we find that the algorithm does not actually pay attention to energy-efficiency from neither energy conservation nor energy fairness points of view. It also bases its operation on the assumption that the sink nodes are stationary (gateway). Local rules and consequently reinforcement paths should be made broader and be based on the node idle time and battery life as opposed to low-delay path in order to address energy efficiency both from an energy conservation and energy fairness point of view. Also the frequency of the refreshment of interests has to be carefully calibrated to ensure the stability of the data delivery as well as the energy efficiency. This technique may be of interest in scenarios when the gateway is fixed.

The GEAR protocol can be useful approach for reaching a region based on a combination of the normalized distance from the closest neighbour to the destination and its normalized remaining energy level. When a packet reaches a target region, recursive geographic forwarding can be more expensive than restricted flooding for disseminating a packet inside the targeted region when network density is low. Also, with low-density networks, some regions can be empty of nodes without the forwarding node being able to realize this. This can lead to large energy waste with nodes neighbouring the empty

region trying to get the packets inside that region unsuccessfully until the TTL count in the packet is exceeded. The authors recommend handling the low-density situation of the network by getting the node to use flooding once it detects that the number of its neighbours is below a certain threshold. This solution in our view is not robust as it assumes uniform distribution of network nodes. In addition to the above issues, energy balancing in GEAR assumes that nodes do not consume energy when in idle mode or while sending or receiving control packets, which is not a reasonable assumption. This affects the usefulness of its energy-efficiency mechanism. We believe that the recursive geographic forwarding is unnecessary when the query is data-centric and needs to be propagated in a certain region. All nodes in the targeted region should receive the query and reply if they have matching interests.

AODV provides a solid technique for handling network operation issues, and is proven to deal with dynamic network topologies as well as nodes mobility quite well. It also handles node and network limited resources sensibly. AODV, however, does not have an inherent energy-efficient strategy as we mentioned earlier. This can be remedied by using an ad-on energy-efficient technique. In addition, the AODV algorithm suffers from two other main issues. First, its routing strategy is centred on finding robust and fast routes between specific sources and destinations, which can be termed as address-centric routing. In the sensor network world, even though the sink (destination) nodes are known and determined upfront in most cases, the source nodes may not be as clear. This is because the required data may come from “any” node in a certain vicinity. However, when established source-destination pairs become known, routing robustness and optimization, a feature of this category, becomes handy. Second, due to the fact that nodes establish routes only on demand, there will be some delay in initial data transfers due to the route establishment delays. This may or may not be an issue depending on the application at hand.

The LEACH algorithm would seem to handle scalability properly from communication volume point of view by dividing the network into clusters with one node acting as a communication representative for each cluster. However, LEACH attempts to handle communications with the sink via a single hop approach. Consequently, each cluster head communicates directly with the sink by adjusting its transmission power. In situations where the network is spread over a large area, this may not work unless a multi-hop communications functionality is added, where the cluster head selection is based on sensor nodes that have more capability and support longer communication ranges (full function devices” (FFD)). Also, with a single node handling communications for the whole cluster at any point of time, this may cause the communications from and to this cluster to be interrupted for a period of time if this node fails, until all nodes within the cluster wake up and decide to elect a new head to recover from this failure. Energy unfairness between cluster nodes at different points of time can result in energy imbalance, which has the effect of shortening the network lifetime. The selection of the cluster head should therefore also take battery life into consideration. Another issue exists which is the complexity and energy cost of the cluster calculation, scheduling and synchronization algorithm.

The proactive grid calculation approach of the TTDD algorithm may have a positive effect in lowering data transfer latency within the network. This, however, comes at a cost in terms of complexity of the algorithm as well as the high communications overhead. There are also some other issues with the TTDD algorithm. First, sub-optimal paths can result from grid based forwarding thus costing more in terms of energy and delay to do the routing over a technique that uses, for example, shortest path routing. Second, the algorithm does not pay attention to energy balance when it selects the nodes that will carry out the forwarding duties. Neither does it attempt to rotate the routing duties between different paths, which results in depleting the energy of some network nodes much faster than others. This becomes more evident when there is a continuous flow of data. Since there is no inherent energy-efficiency capability used by TTDD, the possibility of integrating an ad-on energy-efficient technique needs to be investigated.

Based on this discussion, we conclude that choosing the right algorithm for the operation of sensor networks has to be made based on the following decision elements:

- Best fit for the application at hand and network conditions and particularly for the tunnel scenario,
- Experimental comparisons, and,
- Adaptability of the algorithm to modifications to meet the conditions discussed in the first section.

4.2 Geocasting

In geocasting a source sensor node sends data to a group of receiver sensor nodes which are determined by a geographic region.

Basically geocasting is very similar to multicast although a geographic region is used instead of a multicast address. This region could be, for example, defined by the lines of longitude and the lines of latitude. Here one sensor node joins a "geocasting group" by entering a specified geographic region and leaves a "geocasting group" by leaving this area. A geographic destination address could be represented by some closed polygon such as:

- point
- circle(center point, radius)
- polygon(point(1), point(2), ... , point(n-1), point(n), point(1))

where each vertex of the polygon is represented using the geographic coordinates [70]. This notation could be used to send data to all sensor nodes within the geocasting region defined by the closed polygon. Geocasting assumes that the position of the sensor nodes is determined in advance. This information could be derived through techniques like GPS and others. It is important to stress that not all sensor nodes need to be GPS-capable. The GPS application consumes a lot of energy and is therefore only reasonable on more powerful nodes such as so called clusterheads. The majority of the sensor nodes might use trilateration or triangulation to determine their relative positions. This technique is based on distance measurements to sensor nodes which are aware of their location.

After the above mentioned general viewpoint one can envision the application of geocasting protocols in the road tunnel scenario which is described in chapter 2.

In the normal case periodic environmental measurements such as temperature and pollution could be obtained from various but specified areas, for example, around the endpoints and the center of the tunnel. This information is useful in respect to maintenance and after accidents if it is dangerous for people to leave the car or not. In case of an emergency like fire in the tunnel, fire-trucks and fire-fighters can request information about the region of interest and derive a status report about the health and fire conditions. This information is helpful for emergency staff which does not use breathing protection and need to be close by.

The internet might be the origin of an information request in the normal case and in case of the emergency either the fire-truck or the moving fire-fighter could be the origin. Those cases need to be handled in a flexible way since one faces possibilities with different degree of mobility.

Hence, any origin of the request need to be flexible to address sensors in the specified area by using geocasting protocols and gather the requested status by using the reverse approach.

A review and discussion of below mentioned and other geocasting protocols can be found in [71] and [72].

4.2.1 Geocasting categorites

Geocasting approaches can be separated in two categories: the data transmission based and the routing creation based protocols.

4.2.1.1 Data transmission based protocols

This category uses a sort of flooding to transmit data to the target node and aims at a reliable delivery. In this context reliability means that as much as possible sensor nodes in the geocasting region should be reached. In particular no route will be established to the target region. The main advantage of data transmission based protocols is no overhead due to the maintenance of routes and the high reliability. The main disadvantage is the higher traffic which results in a higher energy consumption of the sensor nodes.

In the following we will give a rough overview of some existing data transmission based geocasting protocols.

4.2.1.1.1 Location based multicast

Location based multicast (LBM) [67] can use two methods. First, one needs to define a forwarding zone in which flooding is used to transmit data packets to the geocasting region. Only sensor nodes within the forwarding zone flood the packets, others do not. One can determine this forwarding zone, for example, as a rectangle which covers the geocasting region and the source sensor node.

Second, the decision if a packet is flooded is not based on a forwarding zone but is based on a distance vector metric in respect to the center of the geocasting region. Thus, whether a packet should be forwarded is just based on the relative distance of the nodes.

The reliability of this approach is slightly less than pure flooding. The main advantage is that the traffic can be reduced significantly.

4.2.1.1.2 GeoGRID

This approach divides the network into a grid [69]. Only one sensor node is supposed to flood a data packet within each square (d times d) of the grid. The election of this particular node called the gateway node is important. The gateway node will not change as long as the gateway node stays within the related square of the grid. If that particular sensor node leaves the square the gateway functionality will be passed to another sensor node within the related square.

This approach reduces traffic significantly and achieves the reliability close to the LBM approach. GeoGRID is preferable compared to LBM especially in dense networks and if applied as Ticket-Based-GeoGRID which achieves the better trade off between traffic and reliability. The philosophy is that each ticket is responsible for carrying one copy of the data packet to the geocasting region.

4.2.1.2 Routing creation based protocols

Protocols of this category first create a route which is then used to transmit data to the target region. The main advantage of this approach is the better energy efficiency due to reduced traffic. Obviously this comes with more overhead due to creation and maintenance of routes.

In the following we will give a rough overview of one existing routing creation based geocasting protocol, namely the GeoTORA approach.

4.2.1.2.1 GeoTORA

This approach is based on the unicast protocol. GeoTORA [68] tries to find redundant routes to the target so that there exist more than one route at the same time which can be used by the source sensor

node. The source sensor node essentially performs anycast to any member of the geocasting region via the TORA protocol.

The advantage of TORA is obviously in case of link failures and network partitioning. GeoTORA is not as reliable as, for example, LBM or pure flooding. At least the traffic is reduced since flooding is used only once at the beginning.

4.3 Conclusions and Future work

In this section, we discuss the conclusions and future work for the two main items covered in this chapter, namely the unicast routing and geocast routing, respectively.

4.3.1 Unicast routing

There exist many choices for routing protocols that can potentially be used in the context of the RUNES project. These protocols belong to different routing strategies and architectures; each of which has emphasis on a specific goal, e.g. energy-efficiency, shortest path, minimum latency, etc.

In this chapter, we discussed the requirements that should be fulfilled by the routing algorithm that will be used within the RUNES project. We also described some of the existing algorithms and shed light on their main strengths and shortcomings. Through this discussion, we indicated that there is no one algorithm that can achieve the required goals and objectives without utilizing additional supporting elements and modifications.

In order to be able to make an informed decision going forward, we intend to run some experimental simulation comparisons between the algorithms that have the best potential. We then intend to use the outcome of this study to guide us as to the best algorithm suited for the RUNES purpose. This will also provide some guidance with regard to the modifications needed to be made to the best algorithm emerging from this study in order to rectify its short comings and make it as compliant as possible with the requirements that we have outlined.

4.3.2 Geocasting

The major challenge in actually enabling geographic addressing and geocasting in real-life sensor and embedded networks is to make the protocols enabling these functions robust to phenomena not included in the mathematical analysis that they typically are based on. For example, several protocols ignore complicated propagation characteristics encountered in wireless channels, assuming bidirectional and circular coverage patterns. The functionality of such protocols can even collapse when used in deployed networks, where substantial deviations from the ideal channel case are often observed.

Second significant item for future work is the reduction of the network protocol maintenance overhead. Several protocols employ periodic beaconing to keep the neighbourhood tables up-to-date, which can cause substantial energy consumption, potentially making the protocols unusable for networks with long deployment times. Beaconless protocols are strong candidates for such applications, and will be further studied and developed in the project.

5 Future in Network Processing Work

In the sensor network domain, one has to face the huge amount of data which is gathered by numerous sensor nodes. In general, routing aims at data transfer between nodes with minimum overhead. Primarily transmission and reception consume most of the energy on the sensor node level. One way to reduce the energy consumption is to reduce the amount of transmissions to the necessary minimum. Thus, in the sensor network context there is a need to reduce the amount of data.

In-network processing, as an integral part of the sensor network paradigm, follows this strong objective. This approach leads to sustainable energy conservation and extended system lifetime due to less necessary transmissions.

We refer to in-network processing in terms of any signal processing inside the sensor network. In more detail, this area covers, for example, data aggregation, data fusion, filtering, mapping and so on. Such techniques are the key aspect of turning data into useful and meaningful information.

Sensor network data from multiple sensors with overlapping sensing regions is always correlated. Hence, different sensors may report the same information in different ways, either in entirely different forms, or with different degrees of accuracy, or some faithfully and some erroneously. Hence, multiple representations of data of the same event can be reduced by local composition of data based on the removal of redundant information. This approach is based on local decisions which are based on application sensitive rules.

Within this discussion one might raise the following questions: At what time, which and how many nodes should collaborate to accomplish the in-network processing task? What kind of data and modalities can be aggregated or fused efficiently? What kind of metrics, criteria and rules can be used while facing well-known constraints? How can one determine the points of aggregation that executes the actual computation under certain requirements such as energy-efficiency and equally distributed energy?

An important issue is the design of an optimal data aggregation schedule that is energy and time-efficient. Data aggregation which can be seen as reverse multicast and data fusion can significantly reduce the transmission overhead within the network where one should be aware of the accuracy, error minimization and loss of information. In most cases the gateway could be a reasonable point of aggregation but in general one should aim at reducing traffic on a level as low as possible in order to achieve the maximum gain. One should be aware of time synchronization that is crucial for reasonable application of above mentioned techniques.

Again clustering approaches seem promising when employed for data aggregation and energy-efficient communication, for example, based on geographic proximity or other criteria such as energy status or sensor modality.

In general in-network processing needs to be designed in conjunction with the routing protocol of the envisioned architecture which is based on the given task, requirements and constraints.

Future work is warranted according to RUNES deliverable 1.2 requirement 43 (the architecture must support the use of flexible data aggregation/fusion techniques) and will include further studies of promising techniques and their applicability in RUNES sensor networks.

6 Autoconfiguration

Before being able to become integrated in an IP network and start IP based communication with peer devices, a certain amount of configuration tasks needs to be completed on an IP node:

- First of all, an IP node needs an IP address, identifying its location within the network. For scalability reasons in most subnets the IP addresses of the different nodes will be aggregated under a certain prefix.
- Secondly, the IP node needs to become aware about the addressing used on its local subnet, that is, to become aware which other IP nodes will be reachable directly on its local interface.
- Thirdly, for communicating with peers outside the local subnet the IP node needs to detect the appropriate gateways to be used for packet forwarding and set the respective routes to them.
- Finally, depending on the application and scenario, an IP node needs to detect other services like DNS servers.

In the PC and server world much of this configuration work today is still be done by humans, either an administrator or the user itself, however, sensor / actuator nodes are embedded devices with limited configuration possibilities via human machine interfaces (HMIs). Furthermore the majority of their users would not have the expertise to perform the required configuration. Consequently one needs to investigate how the required configuration tasks as described above could be handled.

One solution could be to deploy sensor / actuator nodes with manufacturer provided default configuration. This approach would be easy to realize, however, the flexibility to use the sensor / actuator nodes in different scenarios would be limited. Consequently an automatic way for allowing sensor / actuator nodes a self-configuration is extremely important.

6.1 Address autoconfiguration

6.1.1 Link layer address autoconfiguration in IEEE 802.15.4

IEEE 802.15.4 technology is highly interesting as communication technology in RUNES networks, e.g. for interconnecting sensor routing nodes. Therefore in this section we will review IEEE 802.15.4 [4] with respect to its built in address autoconfiguration as well as how the network layer could base its address autoconfiguration on parameters determined during the IEEE 802.15.4 autoconfiguration procedure.

PAN identifier:

An IEEE 802.15.4 network consists of independent Personal Area Networks (PANs). Each independent PAN has one PAN controller and a unique 16 bit PAN identifier. The unique PAN identifier is included in MAC messages and enables, first, communication between different independent PANs and, second, the assignment of 16 bit short addresses (see next section).

The PAN identifier is chosen by the PAN coordinator. Thereby, in order to select a unique identifier, the PAN coordinator could perform an active scan. The active scan results in information about other PANs within range and their respective PAN identifiers. Having a mobile scenario, the set of PANs within range may change over time and PAN identifier conflicts may occur. For this case 802.15.4 supports PAN identifier conflict resolution as follows:

- **Conflict detection:**

  The PAN coordinator determines a PAN identifier conflict in case it receives a beacon frame from another PAN coordinator (PAN coordinator subfield set to 1) with a PAN identifier equal to its own PAN identifier (denoted as macPANId) or in case it receives a PAN ID conflict notification command.

  PAN devices that are no coordinator determine a PAN identifier conflict when receiving a beacon frame from a PAN coordinator with the PAN identifier equal to macPANId but with an address that is not equal to the current PAN coordinator address (macCoordShortAddress or macCoordExtendedAddress).

- **Conflict resolution:**

  After identifying a PAN ID conflict, a PAN coordinator performs an active scan, selects a new PAN ID, and signals the new assignment via broadcasting a Coordinator realignment command into the PAN.

  In case a normal (not a coordinator) device identifies a PAN ID conflict, it sends a PAN ID conflict notification to the coordinator. Afterwards the PAN coordinator performs the resolution procedure as given above.

PAN device addresses:

Each IEEE 802.15.4 device has an IEEE 64 bit extended address (EUI-64 format). The IEEE extended address is a concatenation of the 24-bit company_id value, assigned by the IEEE Registration Authority to a company, and a 40-bit extension identifier, assigned by the respective company [3]. IEEE gives also guidelines for generation of EUI-64 identifiers from 48 bit MAC addresses [3]. This can be used, for instance, to generate a unique EUI-64 identifier for Telos motes from the 48 bit MAC address, stored in write protected flash in a 48-bit silicon serial identification.

IEEE 802.15.4 devices may also use 16 bit short addresses for communication. The allocation is performed by the PAN coordinator upon request during the association phase. Each PAN devices that would like to be part of a specific PAN has to perform the association process. During this process, a PAN device performs an active or passive (no beacon request command) scan. In the Association request command the device can signal the wish to have the coordinator to allocate it a 16 bit short address by having the Allocate address bit set. If the short address mode is supported by the coordinator and the association request is accepted, the PAN coordinator allocates a 16 bit short address for the requesting device and the PAN coordinator sends an Association response command to the requesting device, specifying the association status and the allocated short address. After a successful association the device stores the PAN identifier (macPANId), and the coordinators addresses (macCoordExtendedAddress and macCoordShortAddress).

In IEEE 802.15.4 it is not specified how a PAN coordinator determines the short address allocation for PAN devices and this is left to higher layers.

In case short address mode is supported in the PAN, source and destination short addresses together with source and destination PAN identifiers are used in MAC messages for communication. In some cases not all of these four fields have to be present.

6.1.2 IPv6 address autoconfiguration in 802.15.4 networks

In RUNES we envisage networks consisting of thousands of network nodes (e.g. sensor networks) and these networks may be connected to others via the Internet. Hence, scalability is a major concern. Due

to the huge address space of IPv6, it is promising to have devices that need global IP addressability IPv6-enabled. Therefore, in this section we will investigate activities with a special focus on address autoconfiguration that apply to IPv6 networks that are based on IEEE 802.15.4 link technology.

Adapting IPv6 neighbour discovery and address autoconfiguration for IPv6 over 802.15.4:

The natural way of performing address configuration for IPv6 networks is to make use of the already standardized mechanisms for any type of IPv6 networks. These mechanisms are mainly specified in two RFCs, one concerning IPv6 neighbour discovery in general [18], and another one addressing IPv6 stateless address autoconfiguration specifically [19].

Regarding an 802.15.4 PAN as a single IPv6 subnetwork, from these RFCs the following functionality would be beneficial to support IPv6 address autoconfiguration in 802.15.4 networks:

- IPv6 stateless address autoconfiguration: Here an IPv6 prefix could be assigned to a single PAN identified by a PAN ID. This address prefix could be announced by selected routers throughout the PAN, and will be taken by sensor nodes for composing their IPv6 address by concatenating the distributed IPv6 address prefix and an EUI-64 based address identifier to be derived from the 16-bit or 64-bit L2 address of the sensor node.
- IPv6 duplicate address detection: Having configured an IPv6 address the sensor node has to verify, that this address is really unique, that is, that the EUI-64 based address identifier chosen by the sensor node has not been chosen by another sensor node in the same PAN.

However, using IPv6 stateless address autoconfiguration and duplicate address detection functionality as specified in the respective RFCs also for 802.15.4 PANs would have some significant drawbacks.

All this functionality requires L2 multicast capability for the following tasks:

- For IPv6 stateless address autoconfiguration a router could send the PAN IPv6 prefix periodically within unsolicited router advertisements. These router advertisements are sent to the all-node multicast address.
- Alternatively a sensor node could solicit the sending of router advertisements. In this case the sensor node would have to send a router solicitation to the all-router multicast address. The router would answer with a router advertisement sent directly back to the requesting node.
- Finally, after having configured its IPv6 address, for performing duplicate address detection a sensor node would check its uniqueness by sending a neighbour solicitation using solicited node multicast.

First of all L2 multicast is not supported on 802.15.4 networks. One way to cope with this missing functionality would be to use the L2 broadcast functionality for this purpose, however, the intensive use of broadcast would lead to a significant consumption of bandwidth, processing power and battery on sensor networks, something which has to be limited as much as possible.

For this reason [17] investigates how standard IPv6 neighbour discovery and stateless address autoconfiguration processes can be optimized for sensor networks. The following assumes that ad-hoc routing inside the PAN will be performed on L2.

In this context one option in 802.15.4 networks would be to send IPv6 router solicitation messages only to the PAN coordinator, as most likely this will be the node providing IPv6 router functionality for the PAN. In meshed networks it should be ensured that router solicitations are restricted to the PAN. Similarly also unsolicited router advertisement should be ensured to not be distributed beyond a single PAN. Additionally the transmission timer for periodically sent unsolicited router advertisements should be significantly increased on PANs.

Performing duplicate address detection might also be optimized in PANs. One possibility here could be that each sensor node always sends its first packet via the PAN coordinator, allowing the PAN coordinator to keep track with the L3 to L2 address bindings of all sensor nodes within a PAN. Consequently instead of sending a neighbour solicitation by solicited node multicast, a sensor node can perform duplicate address detection directly with the PAN coordinator, exchanging neighbour solicitation and neighbour advertisement using L2 unicast. This would involve only the sensor nodes on the path to the coordinator in this process. Similarly IPv6 address resolution and IPv6 neighbour unreachability detection can be done directly between the requesting sensor node and the PAN coordinator, limiting thereby packet exchange to L2 unicast.

With the above mentioned modifications to IPv6 neighbour discovery [18] and IPv6 stateless address autoconfiguration [19] one restricts the PAN design with the requirement for co-location of PAN coordinator and IPv6 router functionality, but provides the possibility to make use of a slightly adapted version of IPv6 neighbour discovery and IPv6 stateless address autoconfiguration for IPv6 address autoconfiguration in 802.15.4 networks.

Theoretically a similar message exchange could also be used for IPv4 address configuration, with the differences, that

- the router solicitation and router advertisement messages would need to be used for stateful address assignment by the router, and
- duplicate address detection wouldn't be necessary for stateful address assignment.

Address autoconfiguration in 6lowpan for IPv6 over 802.15.4:

The IETF 6lowpan working group also investigates and specifies possibilities for the operation of IPv6 over 802.15.4 technology. Based on a problem statement, within scope of the working group is the specification of basic packets formats and a sub-IP adaptation layer for the transmission of IPv6 packets over IEEE 802.15.4 WPAN Networks. Furthermore, since IEEE 802.15.4 devices are expected to be deployed in mesh topologies, the group states that it may also work on an informational document that gives information about how to apply existing MANET protocols to LoWPANs (e.g., AODV, OLSR, DYMO, etc.) [6].

Stateless Address Autoconfiguration of 6lowpan-enabled devices is currently specified in an Internet Draft [5]. The process consists of the generation of a 64 bit interface identifier (IID) and a link-local IPv6 address.

In case EUI-64 addresses are used for 802.15.4 communication, the IID of a device is derived from the 802.15.4 device's EUI-64 address (IEEE 64 bit extended address) due to RFC 2464 [7], where it is specified that the IID is formed from the EUI-64 by complementing the "Universal/Local" (U/L) bit, which is the next-to-lowest order bit of the first octet of the EUI-64. A 1 in the (U/L) bit of the IID signals global uniqueness of the IID.

In case 802.15.4 16 bit short addresses are used instead (see 6.1.1 how short addresses are generated), first a 48-bit address is formed by concatenating 32 zero bits with the 16 bit short address. Afterwards, from the 48 bit address the IID is generated due to RFC 2464. Thereby, the first 3 octets of the 48 bit address are set as first 3 bytes of a EUI-64 address, the fourth and fifth octets of the EUI-64 address are set to the fixed value FFFE hexadecimal, and the last three octets of the 48 bit address become the last three octets of the EUI-64. Afterwards, the IID is formed by the EUI-64 address with setting the (U/L) bit to zero since global uniqueness is not given.

Having generated the IID, an IPv6 link layer address for an 802.15.4 interface is formed by appending the IID to the link local prefix FE80::/64.

Having performed this stateless address autoconfiguration process, a device is ready to communicate with other 6lowpan-enabled devices within the same PAN via IPv6 without manual configuration. The communication possibility could be exploited by stateless or stateful configuration procedures to further configure the network, e.g. for configuration of global IPv6 addresses or for prefix and gateway detection. However this is not specified in 6lowpan currently and needs to be addressed in RUNES elsewhere.

6lowpan assumes that a specific PAN represents a specific IPv6 link with a unique prefix. [5] does not specify how to automatically generate a unique link prefix from the unique PAN identifier, but gives one possibility in which a /64 link prefix is built by concatenating the PAN identifier (16 bit) to a /48 prefix. The PAN identifier can be chosen arbitrarily by the PAN coordinator (see section 6.1.1). In RUNES, an IPv6-enabled PAN coordinator could build the link prefix by using the PAN identifier as given above and send this information via Router Advertisements into the IPv6-enabled PAN. IPv6-enabled PAN devices are then able to generate an IPv6 address by appending the prefix with the respective IID. Please note that IEEE 802.15.4 does not support multicast so IPv6 multicast packets like unsolicited Router Advertisements, sent to the IPv6 all-nodes multicast address, have to be sent as 802.15.4 broadcast frames.

Of course, 6lowpan approaches for address autoconfiguration can only be applied for RUNES networks that operate IPv6 over 802.15.4 and other mechanisms (or modified versions) have to be used otherwise.

6.1.3 Address autoconfiguration of MANETs

In several envisaged RUNES scenarios networks are formed in an ad-hoc fashion. In case the network nodes are IP-enabled, the ad-hoc established networks are Mobile Ad-hoc Networks (MANETs) as investigated in the IETF MANET working group.

So far no standard for autoconfiguration of MANETs exists. The IETF MANET working group have generated initial ideas and, after getting aware of its complexity, outsourced the autoconfiguration issue from its charter and formed a new IETF working group denoted as Ad-Hoc Network Autoconfiguration (autoconf).

Like all IP nodes, nodes of a MANET need an IP address configured on its network interface(s) valid within the MANET for intra-MANET communication or valid globally for communication with devices in the Internet. However, traditional protocols used for autoconfiguration of IPv6 nodes like IPv6 Neighbour Discovery (RFC 2461) and IPv6 Stateless Address Autoconfiguration (RFC 2462) expect multicast-capable links, i.e. an IPv6 multicast packet sent to the link is received by each IP node connected to the link directly without the requirement for intermediate nodes performing IP forwarding. Nodes of a MANET do not share access to a single multicast-capable link and other processes have to be used for autoconfiguration. Furthermore, in MANETs no central management entity like a DHCP server may exist or may be reachable. What complicates the address autoconfiguration further is the possibility of MANET partition and mergence due to its mobile characteristic. Therefore, even in case address uniqueness is proven once, duplicate addresses may occur after a mergence of two or more MANET partitions. Therefore, other ways for MANET autoconfiguration have to be developed.

So far in the IETF autoconfiguration working group several individual drafts tackle this issue. Some of these approaches are outlined in the following:

AROD: An address autoconfiguration with Address Reservation and Optimistic Duplicated address detection for mobile ad hoc networks [9]:

AROD propose a distributed address autoconfiguration approach for MANETs using an address reservation mechanism in combination with optimistic Duplicated Address Detection. Thereby, nodes in the MANET exist that have beside their own address a reserved address. A new node joining the MANET selects an agent node and requests the agents reserved address. If the agent has a reserved address it gives it to the new node. Otherwise the agent searches its neighbourhood for nodes with reserved addresses. After successful address allocation, the agent generates two random addresses, checks their uniqueness by performing DAD once, and, if successful, gives one address as reserved address to the new node and keeps the other address itself. The DAD process is characterized as optimistic since it is performed only once, whether successful or not. The AROD mechanism is expected to have low address allocation latency time due to getting quickly a reserved address and low communication overheads due to the optimistic DAD. Especially the latter fact is interesting for resource-constraint RUNES networks. The mechanism is considered for IPv4 and IPv6. So far, only an overview of the mechanism is given and no detailed protocol specification. Moreover, the mechanism does not consider merging of MANETs.

Ad Hoc IP Address Autoconfiguration [8]:

The focus of this approach is to handle MANET partition and mergence and to resolve duplicate addresses. The mechanism is designed for IPv4 and IPv6 and message types for the processes are already specified. The document specifies the process of selecting a random generated address with MANET scope for IPv4 and IPv6, determination of its uniqueness in the MANET, and permanently checking address conflicts due to mergence. Duplicate Address Detection is based on a hybrid mechanism, applying Strong DAD for determining address duplication in a specific partition and Weak DAD for determining duplicate addresses after partitions have merged. Thereby, Strong DAD is based on a request-response protocol where a node that intends to assign a newly generated address to one of its interfaces sends an Address Request (AREQ) message into the MANET and waits for Address Replies (AREPs). When not receiving an AREP message a node concludes address uniqueness. Retries are possible, however, the counters should be set carefully in RUNES networks since each retry causes overhead.

In Weak DAD a combination of IP address and a key is used to determine duplicate addresses. Weak DAD requires an additional "Key" field in routing tables, which means to re-design already existing routing tables and means therefore a lot of effort in the RUNES context. The key value is assigned to each network interface. It is appended to control packets of ad hoc routing protocol packets – such as route discovery and hello messages – and intermediate nodes must maintain the key value for each address in the routing table or cache. Address conflict is determined in case a routing control packet with the Interface-Key extension is received with an IP address that matches an existing routing table or cache entry but with different key values.

Extensible MANET Auto-configuration Protocol (EMAP) [12]:

EMAP is an autoconfiguration protocol for IPv4 and IPv6 MANETs. In isolated MANETs EMAP can be used to autoconfigure a unique (or at least highly likely unique) IP address with MANET scope. In hybrid MANETs (with connection to the Internet) EMAP can even be used to autoconfigure globally routable IP addresses. Moreover, the EMAP framework is designed to support service discovery for MANETs, which will be discussed in section 6.3.

A key point in EMAP is the support for proxies, i.e. a MANET node that is not providing a concrete service but answers requests on behalf a node providing the service.

So far two EMAP message types have been specified: EMAP_REQUEST and EMAP_REPLY. Beside a type field, EMAP messages contain a code field that signals its function, e.g. Duplicate Address Detection (DAD), Global Configuration (GC), and DNS Server (DS) discovery. Depending on its type and functionality, EMAP messages are either "flooded" into the MANET or send via unicast. How flooding is performed is not specified but it is recommended that optimised mechanisms should be used.

Concerning autoconfiguration of a MANET-local IP address (denoted as local configuration), a node selects a temporary address used as Originator Address in DAD request messages and a tentative address that is used as Requested Address in DAD request messages and as Originator Address in DAD reply messages.

For IPv4, both addresses are taken from the 169.254/16 subnet, with the 16 low-order bits chosen randomly in the range of 1 -2047 for temporary address and in the range of 2048 – 65534 for the tentative address. For IPv6, no decision is made yet but an option is given. Thereby, Unique Local Addresses [13] are formed with a special Global ID field reserved for MANET, the Subnet ID chosen equal to the 16 low-order bits in IPv4, and the EUI-64 interface address of the requesting interface used as interface identifier. DAD for the tentative address is performed similar to [8] (see former section).

Concerning Global Configuration (GC), i.e. the autoconfiguration of a globally routable address that allows MANET nodes to communicate with nodes in the Internet, an Internet Gateway (IGW) may flood periodically prefix announcements via GC_REP messages into the MANET or may reply to a GC_REQ message via a unicast GC_REP message. In the GC_REP message the global prefix is sent implicitly since the receiving node concludes the prefix from the Originator Address of the EMAP message (the global address of the IGW) and the Prefix Length field.

In IPv4, a node receiving a GC_REP message appends the advertised prefix by a random number and applies (should) DAD. In IPv6, in case the advertised prefix is lower than 64 bits, a node receiving a GC_REP message generates a 64 bit prefix by appending the advertised prefix with a random number and appends this 64 bit prefix by the EUI-64 interface identifier of the respective network interface; otherwise the global address is formed by appending a random number to the advertised prefix. Afterwards, the node may/should perform DAD.

Currently, EMAP is just specified in an individual draft. Furthermore, the draft states that EMAP should be integrated in MANET routing protocols (e.g. OLSrv2 or DYMO) and leaves the integration specification to MANET routing protocol developers. Moreover, it is not specified how to "flood" EMAP messages. An option would be to apply SMF [14]; however, as with multicast in general, the proxy functionality of EMAP would not be exploited since request messages sent via multicast would be distributed in the network regardless whether intermediate nodes reply on behalf of the final destination. This means that there is still a lot of work to be done to make EMAP become a standard. Nevertheless, EMAP could be used in a pre-standard version in RUNES or at least its principles could be used as basis for a RUNES solution.

6.2 Gateway detection

MANETs may be standalone networks or may be connected to the Internet via an Internet Gateway. Internet Gateways may be mobile or fixed, single or multiple, equipped with wired and/or wireless network interfaces [10]. In order to be able to communicate with nodes in the Internet, in RUNES a MANET node has to perform gateway detection in order to obtain information about which node to send packets destined for the Internet. Additionally the MANETs need to be aware to a certain extent about the addressing concept used, that is, they need to know which address can be reached within the MANET, and for which they need to go via the gateway.

6.2.1 Adapting IPv6 neighbor discovery for IPv6 over 802.15.4

This section assumes that ad-hoc routing inside the PAN will be performed on L2. Performing ad-hoc routing on L3, there might be alternative and more efficient ways to do IPv6 gateway detection by means of the L3 ad-hoc routing protocol than the one described in this section.

The natural way of performing gateway detection for IPv6 networks is to make use of the already standardized mechanisms for any type of IPv6 networks. A commonly used mechanism is specified in IPv6 neighbour discovery [18].

Regarding an 802.15.4 PAN as a single IPv6 subnetwork, the following functionality would be beneficial to support IPv6 gateway detection in 802.15.4 networks:

- IPv6 gateway detection: By periodically sending of IPv6 router advertisements any IPv6 host will learn about several possible gateways located on their respective link. Additionally a host could explicitly trigger a router advertisement by sending a router solicitation message to the all-router multicast address. In this case the router replies with his advertisement to the requesting node's unicast address only.

However, using the IPv6 gateway detection mechanism mentioned above also for 802.15.4 PANs would again require L2 multicast or broadcast capability for sending the router solicitations and advertisements to the all-router respectively all-nodes multicast address. For the well-known reasons of bandwidth, processing power and battery consumption intensive use of L2 multicast or broadcast has to be avoided in 802.15.4 networks.

Similar to using IPv6 neighbour discovery for stateless address autoconfiguration also for IPv6 gateway detection one could avoid sending unsolicited router advertisement, or at least significantly increase their transmission timer on 802.15.4 networks. The main way to detect gateways should be the solicitation of router advertisements by the sensor nodes. However, also in this aspect the sensor nodes should not send their router solicitation by L2 multicast or broadcast, but explicitly to the PAN coordinator, which is likely to be the IPv6 gateway. The PAN coordinator could then reply with a router advertisement sent back directly to the L2 address of the sensor node. This would involve only the sensor nodes on the path to the coordinator in this process.

With the above mentioned modifications to IPv6 neighbour discovery [18] one restricts the PAN design with the requirement for co-location of PAN coordinator and IPv6 router functionality, but provides the possibility to make use of a slightly adapted version of IPv6 neighbour discovery for IPv6 gateway detection in 802.15.4 networks.

Theoretically a similar message exchange could also be used for IPv4 gateway detection.

6.2.2 Global connectivity for IPv6 Mobile Ad Hoc Networks [11]

A gateway detection solution for IPv6 MANETs is specified in [11]. The document describes a mechanism that enables a MANET node to acquire a globally routable address from an Internet Gateway and how the node communicates over the gateway with nodes in the Internet. A MANET node discovers an Internet Gateway by receiving an Internet Gateway advertisement. Internet Gateway advertisements may be advertised periodically or in response to a solicitation. From a received Internet Gateway advertisement a node learns prefixes, configures a routable IP address by using the received prefix in combination with its EUI-64 interface address, and inserts the gateway address as Internet route in its routing table.

Internet-gateway advertisements and solicitation messages may be implemented as modifications to Router Advertisement and Solicitation messages of the IPv6 Neighbour Discovery Protocol (NDP) or as additional control messages of the routing protocol in use. This of course is a critical requirement, requesting for IPv6 implementation or MANET protocol implementation modifications. The document specifies modification of NDP messages but not additional routing protocol messages.

The NDP modification foresees a MANET Router Advertisement/Solicitation Flag that signals MANET nodes that this message can be sent over multiple hops and has not just link-local scope (as usual NDP messages).

6.2.3 Gateway detection functionality in MANET routing protocols

Some MANET routing protocols already specify gateway detection, for instance, OLSR and DYMO described in the following. In IETF RFC 3626 [15] , specifying Optimized Link State Routing (OLSR), already a mechanism has been defined that supports gateway detection via so called Host and Network Association (HNA) messages. A MANET gateway, a node that has beside interfaces towards the MANET (participating in OLSR) also non-MANET interfaces, periodically sends HNA messages into the MANET, containing prefix information of external networks reachable via the MANET. Prefix information could also give the default route to the Internet. From the information given in HNA messages receiving MANET nodes learn the routes towards external networks, i.e. the address of gateways towards the Internet. In [15] only the IPv4 packet format of HNA messages has been specified. However, there are already implementations for IPv6-enabled HNA processing available, e.g. at www.olsr.org.

In DYMO [16], a gateway node that is connected to the Internet responds to Route Request (RREQ) messages with a TargetAddress outside its configured MANET with a Route Reply (RREP) message. Thereby, the gateway should set the G bit in any Route Element (RE) sent or processed since the G-bit flag indicates to nodes in the MANET that the RBNodeAddress (belonging to the gateway) is attached to the Internet and is capable of routing data packets to all nodes outside of the configured MANET subnet.

6.2.4 Gateway detection in EMAP [12]

In the EMAP protocol (see section 6.1.3 for a more detailed explanation) gateway detection is realized similar as in [11]. An Internet Gateway sends either periodically or upon request an announcement message into the MANET, specifying a global prefix (concluded from Originator Address and Prefix Length) and the Internet gateway address. In case a node receives Internet gateway announcements from several gateways, a MANET node should select one as default route to the Internet. Selection could be based on the Distance field in GC_REP messages, which is incremented in each MANET node before being forwarded.

Regarding usability of EMAP in RUNES the same aspects as stated in the section dealing with EMAP for address autoconfiguration apply.

6.3 Service discovery

Having obtained IP addresses automatically (see section 6.1) and detected a gateway to the Internet (see section 6.3), a MANET node may be interested to autoconfigure additional services, e.g. the IP addresses of DNS servers. In the RUNES context, the more configuration can be applied automatically the less expertise is required from personnel that establishes networks in the field in an ad hoc manner. Which service is required and, hence, which service has to be discovered is clearly scenario and

application dependent. Nevertheless, in RUNES at least the possibility for service discovery should be developed and implemented.

Service discovery in EMAP [12]:

As mentioned in previous sections EMAP is an autoconfiguration protocol for IPv4 and IPv6 MANETs that supports service discovery. In the current Internet Draft, only discovery of DNS servers is specified, but due to its extensibility, the protocol can be enhanced with support for discovery of other services as well. For service discovery, nodes that know the address of entities that provide services (e.g. a DNS server) either periodically advertise this information to the MANET or respond to an explicit client request. The advertising node may provide the service itself or just know the address of the respective server and act as kind of proxy for autoconfiguration.

For DNS server (DS) discovery and configuration, an entity called DNS Server Advertiser (DSA) provides the IP address of a primary and perhaps of a secondary DNS server. The DSA may be collocated with an Internet gateway (IGW). Similar to gateway detection, a DSA may flood a DS_REP into the MANET periodically and send a DS_REP message as response to a DS_REQ message from a client. DS_REP messages contain the IP address (IPv4 and/or IPv6) of a primary and of a secondary DNS server.

Regarding usability of EMAP in RUNES, the same aspects as mentioned in previous sections apply.

6.4 Conclusions and Future work

In RUNES networks autoconfiguration of network nodes is highly desirable since nodes may provide no or only limited human machine interfaces, networks may scale to thousands of nodes, and RUNES networks are likely to be deployed in an ad hoc manner with no pre-existing infrastructure at all.

In IP networks, autoconfiguration is required to select and configure an IP address (IPv4 and/or IPv6) with subnet scope or even a globally routable IP address in case communication to nodes in the Internet is required. Moreover, detection and configuration of gateway addresses that route packets to the Internet is required and, depending on the application and scenario, also the discovery and configuration of other services like the addresses of DNS servers is needed.

A lot of work has been performed in the area of autoconfiguration but, especially in the field of ad hoc networks, no standardized mechanism exists so far and it is unlikely that a standardized mechanism will come up soon that could be deployed in RUNES. Nevertheless, some approaches in the research community, especially in IETF working groups, are promising and could be adapted, extended, and combined to a RUNES approach.

For address autoconfiguration of RUNES networks that use IPv6 over IEEE 802.15.4 the autoconfiguration mechanisms specified in the IETF working group 6lowpan could be used (see section 6.1.2) to automatically configure a link-local IPv6 address and the combination of PAN ID and Interface ID could be used to generate an IPv6 address with IEEE 802.15.4 network scope (not specified yet). Alternatively a slightly adapted version of IPv6 neighbour discovery and IPv6 stateless address autoconfiguration protocols could be used.

In RUNES link technologies beside 802.15.4 are also within scope. Especially in the IETF autoconf working group development of autoconfiguration mechanisms that are applicable for mobile ad hoc networks deploying any type of link technologies are ongoing.

For address autoconfiguration, some approaches select an IP address with a randomly generated part and perform afterwards duplicate address detection ([8], [9], and [12]). Ad Hoc IP Address

Autoconfiguration [8] even supports the merging of MANETs. All investigated approaches support IPv4 and IPv6. Only EMAP deals explicitly with configuration of a global routable IP address.

Gateway detection is usually performed via a message, containing information about the external network prefixes and the gateway address, which is send either periodically or upon request into the MANET. Some MANET protocols (OLSR and DYMO) have gateway detection built-in, other approaches exist that needs integration in NDP [11] or into MANET protocols [12]. Alternatively a slightly adapted version of IPv6 neighbour discovery could be used. For resource constraint RUNES environments, clearly explicit request are preferable since consuming less network resources.

Another interesting autoconfiguration protocol is EMAP since it features address autoconfiguration (with MANET scope as well as global scope), gateway detection, and service discovery for IPv4 as well as IPv6. However, EMAP is expected to be integrated in existing MANET protocols. Since RUNES technology is not limited to a specific MANET protocol, this means a requirement for specification of EMAP integration into various MANET protocols, e.g. into OLSR, AODV, and DYMO. Hence, a common protocol that can be deployed for MANET autoconfiguration independent of the MANET protocol in use seems to be more appropriate for RUNES.

In summary, all of the described approaches are still under investigation; therefore at the time of this writing there has been no strong indication about which of them will be adopted for standardization and which one to deploy best in RUNES. This requires further investigation, development, and testing work to prepare RUNES networks for autoconfiguration.

7 Localization for Nodes in RUNES

In ubiquitous embedded systems, location awareness is essential. First, sensor nodes in a ubiquitous embedded system need to identify the locations where the sensor readings are originated. Second, location information is required by the location-based routing protocols that are often employed in the wireless sensor network which assumes a critical part of the ubiquitous embedded system. Location-based routing protocols use geographical area information to route through large numbers of sensor nodes that cannot be conveniently identified by their individual global node ID. Such protocol aims to save energy and resource by eliminating the need for route discovery and maintenance of large routing tables. Third, for "context-aware" applications in the ubiquitous embedded systems, one of the important dimensions of context is location. Availability of location information in the networks enables the location-aware services and improves application performance. This is also true for application components to improve their efficiency, e.g., in the caching behaviour, in the request-response selection, etc. In addition, security can also be enhanced by location awareness, e.g., detecting and preventing wormhole attacks.

This section analyses the localization requirements in the RUNES architecture, reviews the state-of-art of sensor network localization techniques, and proposes the selected mechanisms that are identified as candidates for support of the RUNES application scenarios. The objective is to propose a localization mechanism that can be effective in most of the RUNES applications. The localization mechanism sought after may involve several different positioning schemes that can be activated under different circumstance. The mechanism would leverage the hardware capabilities of the selected sensor nodes and apply the component-based software positioning capabilities designed to achieve the required localization function.

7.1 Requirements for Localization in RUNES

7.1.1 RUNES Nodes for Localization

In networks, location information is often easily available on some nodes but not others. For nodes with fixed location or adequate hardware, gathering location information is often not an issue. In the general topology of RUNES, the gateway nodes, the WAN nodes and the equipments in the Internet area may usually have no issues in obtaining node location information. These nodes are either fixed, or with GPS [39] capabilities installed. One of the possible localization issues for these nodes may arise when a mobile gateway roams into an area where temporarily, its GPS system cannot function properly (e.g., indoor, under water or in very cluttered urban area), or when some small/economic/ backup mobile gateway nodes are deployed that have no GPS capability. We expect two kinds of localization problems that would need to be solved in the RUNE networking scenarios. One is of obtaining location information in the sensor/actuator network where only limited hardware resource is available due to the node cost, the size of the sensor node and the number of the nodes to be deployed. The second case is to drive the location information for the mobile nodes, for example, of the first responders to a tunnel fire where their GPS devices even if available are not properly functioning in the under cover or indoor surroundings.

From the perspective of obtaining their location information, we categorize the nodes involved in the RUNES scenario as follows:

1. Anchor or Fixed sensor nodes – These nodes either have hardware capability such as GPS to sense its location, or are deployed onto the planned positions though the node should automatically sense and calculate its location with only minimum manual configuration. For example, a node placed in a given room or a section of the tunnel may have its address/node

ID configured so that it can be mapped directly to its location. When the node is activated, it should be able to report its location by mapping its address/node ID. The location information obtained by the fixed sensor nodes may have small errors, or may be only in the format of its relative location, e.g., we know the node in a room rather than its coordinates. This type of nodes when used to facilitate other sensor nodes to obtain their location information, are often called anchor nodes.

2. Ad hoc sensor nodes – These nodes are deployed on the fly to random positions in a given range/region and required to obtain their location in a timely manner. These nodes are mostly static. As an example, when a fire emergency took place in a tunnel, the rescue personnel may start to deploy special sensor nodes while they enter the scene. These sensor nodes cannot be accurately deployed to certain positions due to the dynamic situation and difficulties of obtaining information on the scene. However, the rescue mission needs to rely on these sensor nodes to report the developing situation in the tunnel. These sensor nodes should be able to quickly calculate their positions and report location based readings.
3. Mobile sensor agents – They are only in limited numbers. Each of the agents tracks its positions continuously even when its GPS device fails. These can be the rescue personals or the mobile robots entering the rescue scene for special tasks. They are mobile so that their location needs to be updated on the move. These agents communicate with the sensor nodes. The agents gather information of the situation and report them back with associated location information.

7.1.2 Requirements for Localization Solutions

The following criteria are applied on selecting localization solutions.

4. Scenario requirements

The scenario requirements include considering if we have outdoor/indoor environments and their surrounding conditions, e.g., sunlight exposures, urban or rural areas etc.

In RUNES, various application scenarios are considered, ranging from the smart medical care centre to the rescue missions. Therefore though the outdoor scenario is a given, the cases of indoor, cluttered urban and heavily tree-ed/covered complex terrain are to be included as well. There is no guarantee for line of sight in all the scenarios, nor sunlight exposure can be avoided at all times. In this document, we are focused on the rescue missions where the fire in the tunnel scenario is a typical example. This scenario is particular interesting and challenging, as it is neither a real outdoor nor a real indoor environment.

These various scenario types [45] affect the selection of the localization approach, especially the ranging technology that is applied in the rang-based localization mechanisms, as elaborated later in the section.

5. Architectural requirements

Localization solutions often have either distributed or centralized architecture, which is very much related to the application architecture. In centralized localization schemes, sensor nodes transmitting data to a central location, where computation is performed to determine the location of each node. Distributed localization methods rely on each node to determine its location using only limited communications with nearby nodes.

The advantages of a centralized localization scheme include that it imposes no computation requirements on sensor nodes, and it is quite suitable for certain application scenarios. The

centralized scheme also has some apparent disadvantages such as a major limitation on scalability due to computational complexity (close to O(n3)). They often scale to a few hundred nodes [36][37][38]. In a Disruption Tolerant Networking (DTN) environment, network transmission cannot always be guaranteed. Centralized location computation often requires considerable bandwidth resource to gather data from all the nodes and also depends on the availability of the network to complete the locating task. During the network partitioning or transmission interruption, location information can be severely delayed or even not gathered. The anisotropic network topology may also impair the algorithm.

Thus, in RUNES, a node should be able to compute its location information in a distributed manner. The disadvantages and issues with the distributed localization schemes are elaborated later in more details. The requirement for distributed location resolution not withstanding; there are many applications where the location information of an individual node is more important to a server. This requires that certain servers should be able to obtain the location information of a remote node. In this case, computing the location information at each node or at a centralized server should be compared using the performance evaluation metrics as described later.

6. Hardware capability of the different nodes in the RUNES network

The hardware capability of the different nodes involved in the RUNES network needs to be identified, as localization relies on the hardware function a node can have. A node with GPS function would have location information available, at least sometimes. A node with certain types of signals and ranging capabilities may apply some localization techniques while other nodes cannot. The memory, computing and communication capabilities of the nodes also affect the choices of the localization scheme.

7. Performance requirements

Accuracy and precision requirements

The selected solution should be able to provide absolute and relative position readings when required by the application. An example of the absolute position reading would be the coordinates with latitude and longitude [59]. A relative position can be, for example, the distance from the south end of the tunnel. The location results can be in the format of measurements, for example, coordinates, distance, angles, etc. The location may also need to be presented in a format of boundaries, for example, in which room, or in which section of the tunnel [62]. In the format of measurements, the accuracy is often required at about half of the radio radius, i.e., 0.5R for 95%-100% percent time of location availability [57]. In our scenario, even if the sensing technology applied would have a larger radius, the requirement for accuracy should be within 10-20 meters in outdoor situation and about 5 meters in indoor cases, considering the indoor or covered areas may have very low visibilities such as in the fire in a tunnel scenario. For boundary locations, close to 100% precision is needed for 95%-100% percent time of location availability.

Scalability

We currently consider the number of sensor nodes in the order of thousands or ten's of thousands. The solution should have the potential to scale when the sensor network goes much beyond this size.

Responsiveness

Responsiveness is measured as the computation time for obtaining the location reading of the node [49]. For any nodes, the latency in obtaining the location reading should not affect the function of the component that utilizes this location information. For example, location based

routing should always have location readings of a node ready when the location of the node is needed. This often entails to acquire the location information ready within the first half minute after a static node is initialised, and to keep up the location update rate of a mobile node up with its moving speed so that the accuracy requirements can be achieved at all times.

Overhead

The overhead of the localization scheme needs to be evaluated, including the hardware overhead, the computational overhead and the bandwidth overhead. The overhead should be minimized and contained within the acceptable level.

Privacy

Privacy is not a major consideration in the first phase of the solution design. However, privacy is an important issue to be considered in the complete solution offer.

In different application scenarios, the above requirements need to be clarified for the particular scenario.

7.2 Related Work

The location information may be required for either a small and static sensor node or a more capable mobile node. There are often two major categories of solutions when computing the node's current location. The first category is named here as sensing based solution, which is for a node to locate itself through signalling and communicating with other nodes. In this type of sensing based solutions, the node that needs to locate itself always has to perform signal exchange with other nodes, which often acting as the reference points. For example, GPS [39] is a typical sensing based solution that a node would exchange signals with several satellites to compute its location. The second type of solutions is often called navigation, or DR (Dead Reckoning) [51]. A DR system needs to know the start point of the node's position. Then by measuring the speed, the direction and the time of the node's movement, the DR system estimates the current location of the node compared with its start position.

7.2.1 DR for Localization

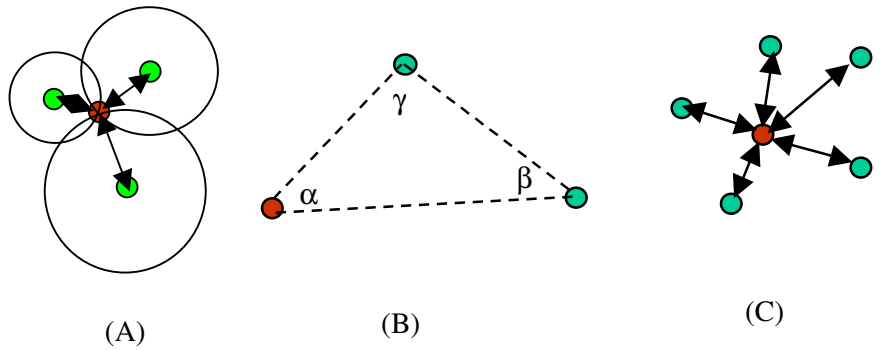
A DR system usually integrates some or all of the following, including the tilt-compensated magnetic compass, electronic pedometer, barometric altimeter, gyroscope and accelerometer etc., to provide a continuous deduced position. Advanced DR system also incorporates GPS module to correct the position calculations when the GPS signal is available. A DR system does not need to communicate or signal with any reference points or nodes to obtain the current location readings. As pointed out in the beginning, the DR system would require a known starting position, and often works well with a map of the area in navigation tasks [52][53]. DR system is often used to track the movement or facilitate the navigation of a mobile agent that can be a rescue/law enforcement/military personnel or vehicle, around city blocks, in indoor/building environments, in heavily tree-ed outdoor environments, in caves etc. DR system has been applied in many scenarios that are quite similar to RUNES scenarios, e.g., the fire in the tunnel scenario where the mobile sensor agent nodes such as the robots or rescue personnel/vehicles are involved.

Though DR system is quite effective in tracking the location and facilitating the navigation of a mobile node in certain application scenarios, there are often several issues associated with the DR system. DR location estimation often accumulates errors through time (3-5 meters in every 100 meters). DR is often not accurate when used in complex terrain. In fire or rescue missions, handling of irregular walking (walk or running of a fire fighter for example) in the step counting algorithm of a DR device is unknown [54]. The second issue is the lack of shared situational awareness. DR system

is a local system that does not communicate with other nodes or references. Though the mobile node that is equipped with a DR may know where it is, this information is not available to anyone else unless transmitted through a network. In a DR system, the current location is obtained with regard to the start point, not to other reference points unless a map consolidates all the location information. In terms of the cost, DR systems have become more economical in recent years, though it is still not a device for a cheap small sensor node.

7.2.2 Sensing based Localization Solution

The sensing based localization mechanism can be further categorized into range-based and range free solutions [46][61]. In either case, the location of coordinates for a node can often be calculated using one of the following three approaches [40] as show in the Figure 1 in a two dimensional presentation. The concept is similar for the 3-D cases.



(A) – Hyperbolic trilateration
(B) - Triangulation
(C) - Multilateration

Figure 10: Locating the Node

In trilateration, the distances from the node to the anchor nodes are measured or estimated. The location of the node is then calculated as the intersection point of three circles to obtain its coordinates. In triangulation, the trigonometric functions are applied to calculate the coordinates of the node from the measured three angles formed by the node and at least two anchor nodes. In multilateration [48], more distance measurements are taken from the node to more anchor nodes to minimize the errors on the resulting coordinates by applying techniques such as the least square on the mean error values etc.

In general, three steps are involved in localization of a node in sensing based solutions:

Step 1 – Constraints establishments

In this step, either through measurements or estimations, certain constraints on the location of the node are obtained. The constraints can be measured values such as distance or angles to several anchor nodes. The constraints can also be estimates of distances through counting hops, observing its connectivity with other nodes etc. The constraints can also be an evaluation on the topology to derive that the current node position must be within a certain region marked by other reference/anchor nodes.

The first step is often communication bounded. In an anchor node based system, this is often to take the measurement between the anchor node and the node that is deriving its own location. For a node to be in the direct range of several anchor nodes, the density of the anchor

nodes may be quite high [56]. When a node has to reach anchor nodes in multiple hops, one of the approaches often applied is flooding [58][35]. Flooding always implies the scaling problem when the network size increases. Fortunately, a good position can be derived in step 2 with knowledge (position and distance) from a limited number of anchors. Therefore nodes can simply stop forwarding information when enough anchors have been "located". This simple optimisation was proven to be highly effective [50].

Step 2 – Location estimates

In this step, algorithms are applied to compute the location of the node. The techniques of computation include the triangulation [42], trilateration [39], proximity detection [57], scene analysis [61], min-mean, least square, etc.

Step 3 – Refinement [35]

This step is often optional. It intends to optimise the results obtained from step 2. Fusion [60] and filtering techniques are often applied at this step to refine the final location readings.

Using different approaches in the step one as described above to establish the constraints, we have then the range-based and range free localization schemes.

In range-based solutions, an absolute point-to-point distance or angle measurement is obtained in the step one and used in the position calculation of step two. The range free technology, on the other hand, does not apply any absolute distance or angle measurement values in step one for calculating the node's position in step two.

7.2.3 Range-based Sensor Localization

Typical range-based solutions include using Time of Arrival (TOA), TDOA (Time Difference Of Arrival) [41] and AOA (Angle or Arrival) [41] measured on a certain received ranging signal, e.g., ultrasound [41], RF [43][44], IR (InfraRed) signal, etc., and RSSI (Radio Signal Strength indicator) [41] measured on the received radio signal to establish the constraints of either distances or angles with the reference nodes, which have known locations. The advantage of range-based solution is their fine resolution often in the results. But ranging techniques are very dependent on the environment, and often require expensive hardware. For example, TDOA often require ultrasound device, and antenna arrays for AOA. In RSSI cases, the results depend heavily upon the assumptions about signal propagation that may not be realistic. In fact, RSSI readings are often found not accurate in estimating the distance [40]. Range errors, if occurs also have hard impact on the position calculation as they can be accumulated.

In Cricket location system [41], a node X in a building uses ultra-sound and RF radio signalling to compute its distance to a heard sensor anchor node though measuring TDOA. Node X would try to obtain such distance information to more sensors anchor nodes that it can hear. Querying a map DB about the radio signal pattern of all the pre-installed sensor anchor nodes, the node X would have a certain idea of how far it is from the heard sensor node. If node X can acquire such distance measurements to at least three sensor anchor nodes, it would be able to compute its location coordinates.

In the Cricket system, large numbers of sensor nodes are pre-installed with planned positions in the building. In other distributed solutions [35][58], fewer anchors nodes with known positions are assumed. Then flooding, or limited flooding is often applied to let more sensor nodes hear about the position information of the anchors and to measure its distance or angles with the anchor nodes. In the flooding, anchor nodes send out its location information in RF signals. Each sensor node receives the

message would use the signal strength (RSSI) measurement, to derive the distance between itself and the sender. This distance information is added to the message before the sensor node forwards it, if forwarding should be done based on the rules of limited flooding as described above, i.e., enough number of anchor nodes have already been heard. In this way the node that is more than one hop away from the anchor nodes can also estimate its distance from the anchor node. When knowing distances from three anchor nodes, triangulation is used to compute the position of the sensor node. It should be noted that the propagated distance is not the real Euclidean distance between the anchor node and the sensor node, as the propagation path may not be a straight line so that the distances can be added. This Distance Vector (DV) distance [58] approach has the advantage of requiring fewer anchor nodes but has the issue with the accuracy. The range errors of the distance on each hop are also easily accumulated to impair the algorithm.

The DV-Euclidean [58] approach on the other hand applies a calculation model of only single hop away offers a higher accuracy of the location calculation. In this approach, signals are also flooded with limit through the network. The nodes that are only one hop away to the anchor nodes would first calculate their locations. Then the nodes that are one hop away from the nodes that have already obtained their locations would calculate, etc. As in the DV-distance approach, the signalling strength (RSSI) is used to measure the one hop distance.

The DV-Euclidean approach yields higher accuracy if the ratio of anchor nodes is high enough, i.e., >20% of the total number of nodes. Euclidean approach also sees less message overhead than the DV-distance approach. However, the number of anchor nodes need to reach 40% of the total number of nodes to ensure all the nodes can deduce its location using Euclidean method. DV-distance approach requires fewer number of anchor nodes to have a complete coverage for all the sensor nodes to locate themselves. But it has higher message overhead, and is susceptible to per hop range errors. DV-distance also does not perform well in the anisotropic network topologies. It may only be applicable in the isotropic topologies.

The effectiveness of the range technology is very dependent on the environment of the application scenario [56]. For achieving high accuracy, ultrasound (e.g., 40khz) and UWB technologies for indoor location sensing would be of good choice. However, none of them are suitable for outdoor environment. The IR-based solutions perform quite well in indoor environments, because IR range is fairly small and can be limited to the logical boundaries of a region, such as a room (by walls). The short range of IR, which facilitates location, is also a major drawback of these systems because the building has to be wired with a significant number of sensors [56]. In the few places where such systems have been physically deployed, sensors have been physically wired in every room of the building. Such a system scales poorly, and incurs significant installation, configuration, and maintenance costs. IR tends to perform poorly in the presence of direct sunlight and hence cannot be used outdoors. Another drawback is that it is a tracking system rather than a self-localization system.

RF radio is often used as an effective ranging technology in the outdoor environment. However, RF cannot be as effectively applied in indoor environments, because radio propagation in indoor environments suffers from severe multi-path effects that make it impossible to precisely control the radio range. One way to solve it is to take pre-measurements of signalling strength of a pre-installed environment [43][44]. Then a signal pattern map is generated that maps a given signal strength pattern to a know location/location area. This map is queried to obtain the location rather than using other signal strength models in the measurement.

7.2.4 Range-free Sensor Localization

Different from the range-based solutions, the range free technology does not attain any absolute distance or angle measurement values in step one for calculating the node's position in step two. Thus it does not use values but employing the estimated region or area as constraints. The range-free

approach thus takes less hardware cost and less (or even none) planning of the system. These approaches are also less subject to measurement errors. However, range-free solutions are often less accurate and incur possibly higher computational overhead.

The range-free localization applies local estimation [56][57] or hop count based estimation techniques [58].

In [55], anchors beacon their position to neighbours that keep an account of all received beacons. Using this proximity information, a simple centroid model is applied to estimate the listening nodes' location. That is, the listener's location is estimated as at the centre of all the heard anchor nodes. This centroid algorithm is simple but would require certain number of anchor nodes to achieve high level of accuracy. The APIT method [57] isolates the environment into triangular regions between anchor nodes, and uses a grid algorithm to calculate the maximum area in which a node will likely reside. APIT either requires the anchor nodes to have higher transmission range than other nodes to achieve the coverage, or would require also a high density of anchor nodes in the network. In Sextant [83], Béziers curves are used to establish the non-convex constraints on the area that a node resides using both the positive and negative information based on what the node can hear in its neighbourhood.

7.2.5 Localization and Tracking of Mobile Nodes

In addition to the DR systems that track the current position of a mobile node by where it is about in reference to its initial point, the sensing based solutions can also and are often applied to calculate and update the position information of a moving node. Such a sensing based solution can often correct the errors accumulated by the DR system, draw relative positions among different mobile nodes instead of knowing only the isolated information of one node as given by a DR system, and optimise the location information for the applications. In this situation, the anchor nodes, that is, the nodes that are used for sensing can be either static or mobile [61]. The nodes that need to be located are mobile. Such a problem can often be modelled using the state-space approach. A powerful statistical tool can be taken here from the Bayesian-filter techniques [86]. Bayesian filters probabilistically estimate a dynamic system's state from noisy measurements/observations of multiple sensors. In the model, the positions of the moving node construct the states of the node. The noisy measurements are the sensing results obtained by the node interacting/sensing with multiple sensor/anchor nodes. Denote the state/position of the node at time t by random variables x_t, and the sensor measurements as a sequence of time-indexed observations of z_1, z_2,…, z_t . At each point of time, a probability distribution over x_t, called "*belief*", $Bel(x_t)$, represents the uncertainty of the estimated position of the node. The $Bel(x_t)$ is defined by the posterior density over the random variable x_t conditioned on all sensor data available at time t:

$$Bel(x_t) = p(x_t \mid z_1, z_2, \ldots, z_t) \qquad (1)$$

The above formula (1) states the probability that the person is at location x if the history of sensor measurements is z_1, z_2, …, z_t . To make the computation tractable, Bayesian filters assume the system is Markov, i.e., $p(x_t \mid x_1, x_2, \ldots, x_{t-1}) = p(x_t \mid x_{t-1})$. Thus, when a new measurement of z_t is obtained, the filters first predict the state at time t according to:

$$Bel^{-}(x_t) \leftarrow \int p(x_t \mid x_{t-1}) Bel(x_{t-1}) dx_{t-1} \qquad (2)$$

Here the $p(x_t \mid x_{t-1})$ describes the system dynamics of its state changing over the time. The filter then corrects the predicted estimate using the sensor measurement of z_t:

$$Bel(x_t) \leftarrow \alpha_t p(z_t \mid x_t) Bel^{-}(x_t) \qquad (3)$$

where the $p(z_t \mid x_t)$ is the perceptual model that describes the likelihood of making observation z_t given that the node is at location x_t.

The implementation of Bayesian filters required specifying the perceptual model $p(z_t \mid x_t)$, the dynamics $p(x_t \mid x_{t-1})$, and the representation of the belief $Bel(x_t)$. When assuming the initial uncertainty is Gaussian and the observation model and system dynamics are linear functions of the state, Kalman filters are the most widely used variant of Bayesian filters and offer the optimal algorithm. If the state space is discrete and consists of a finite number of states, the grid-based methods provide the optimal recursion of the belief. Thus the grid-based approach has the advantage over the Kalman filters as that they can represent arbitrary distributions over the discrete state space. However, the grid-based solution does have the issue of the computational and space complexity required to keep the position grid in memory, and the complexity grows exponentially with the number of dimensions.

In the real cases where the distribution functions are often non-linear/non-Gaussian, the extended Kalman filters, the approximate grid-based methods and the Particle Filters have been developed [84][85][86]. The Particle Filters have especially draw much attention due to their computational efficiency. The Particle Filters apply the SIS (sequential Importance Sampling) algorithm which is a Monte Carlo (MC) method [87]. The key idea is to represent the required belief $Bel(x_t)$ by a set of random samples with associated weights and to compute estimates based on these samples and weights. As the number of samples becomes very large, this MC characterization becomes an equivalent representation to the usual functional description of the belief pdf (the posterior pdf), and the SIS filter approaches the optimal Bayesian estimate. To develop the details of the Particle Filters algorithm, let $\{x_t^i, w_t^i\}_{i=1}^N$ denote a random measure that characterizes the belief $Bel(x_t)$, where $\{x_t^i, i = 0,.., N\}$ is a set of support points with associated weights $\{w_t^i, i = 1,..., N\}$. The weights are normalized such that $\sum_i w_t^i = 1$. Then the belief is represented as

$$Bel(x_t) \approx \sum_{i=1}^{N} w_t^i \delta(x_t - x_t^i) \qquad (4)$$

where $\delta()$ is the Dirac delta measure. Therefore, based on the reported sensing results from sensors, at time *t*, weights may be assigned to different states (positions) [84]. Particle Filters would then update the position at time *t* recursively based on the continuous reporting from the sensors. Particle Filters often achieve good level of accuracy, robustness and are relatively efficient in computation and fairly feasible in implementation [86]. Particle Filters techniques are selected as a candidate solution option for estimation and update of the locations of mobile nodes in RUNES.

7.3 Conclusions and Future Work

In RUNES project, two types of localization problems are identified, as described in the previous sections. The first problem is to locate the static sensor nodes. The pre-installed sensor nodes in the tunnel or building may have known locations that can be programmed mapping to their node ID etc. Thus upon its initialisation, the node can come up with its location information. These are anchor nodes. There are other randomly deployed ad hoc sensor nodes, for example, the special sensors deployed by the fire first responders when entering the tunnel that is on fire. These sensors need to locate themselves in time to report readings associated with their location, and to assist in tracking the mobile agents, including robots, fire fighters and rescue vehicles. Therefore, our first problem is to locate the ad hoc sensor nodes. The second issue as described before is to locate the mobile agent nodes.

The approach under investigation is to apply RF signal sensing to establish the location information of the ad hoc sensor nodes. Though RF sensing may not be accurate in an indoor or tunnel environment, pre-planning can often establish the signal patterns for location references. We also would evaluate the

UWB, Ultra-sound and IR sensing and badge technology to identify if they are applicable in the tunnel environment where the end of the tunnel would be much similar to an outdoor atmosphere with possibly direct exposure to sunlight. RF sensing is firstly considered because it is ubiquitous on the current sensor, sensor routing and gateway nodes. Some of the sensor nodes, both the anchor and ad hoc ones, may come with RFID signal sensing capability. For the mobile agent node, the RF sensing signals are also applied between the mobile agent node and the sensor node (the anchor nodes and other ad hoc nodes that have already established their location readings), or between the mobile agent nodes to establish the relative location with the reference points, in addition to information obtained from possibly the DR system. Particle Filters solution techniques are to be applied in optimising the output location results with multiple inputs from the RF sensing and the DR systems. Such a hybrid solution is expected to outperform either the sensing based solution or an isolated DR system.

8 RUNES Sensor Network Connectivity and Topology

Based on the recent progress of sensor network research and development we can argue that future sensor networks will be heterogeneous in terms of devices and type of sub-networks as well [21][22]. Sensor networks will consist of basic sensor nodes and nodes with significantly higher communication capacity and energy sources (called as masters in [22]) compared to the basic sensor nodes. Such networks (like in **Error! Reference source not found.**) will have two advantages compared to flat networks. First, overall capacity of the network will be improved by the more powerful radios of some special, "enhanced" nodes. Second, the network will be hierarchical, thus it will be easier to manage.

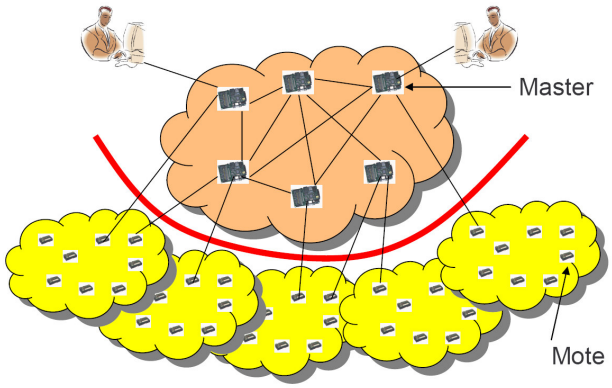


Figure 11: A two-tiered sensor network [22]

8.1 Ad hoc and sensor network connectivity

Basically, two different classes of solutions can be designed how to integrate different types of networks. Solutions in the first class will have nodes with different access technologies, and nodes with the same type of access technology will form one network. Thus, the sensor network will be a set of different sub-networks (e.g., an ad hoc network and a sensor network). Communication between nodes in different networks will be through gateway nodes that have at least two access technologies. Solutions in the second class will contain nodes with different access technologies; however, all the nodes will form one global network independent of the access technology of the nodes. In fact, this type of network can be regarded as an overlay network.

8.1.1 Different access technologies, different networks

An example for sensor networks that is composed of two different subnetworks is the ART-WiSe sensor network [23] designed by the IPP-HURRAY! Research group. They propose a cluster-based two-tiered sensor network composed of two interoperable networks arrayed in a hierarchical manner (**Error! Reference source not found.**). The Tier-1 network is a IEEE 802.15.4-compliant wireless sensor network interacting with the physical environment, while the Tier-2 wireless network is a backbone for the underlying sensor network. Nodes in the Tier-1 network are partitioned into clusters, and each cluster is managed by a node of the Tier-2 network.

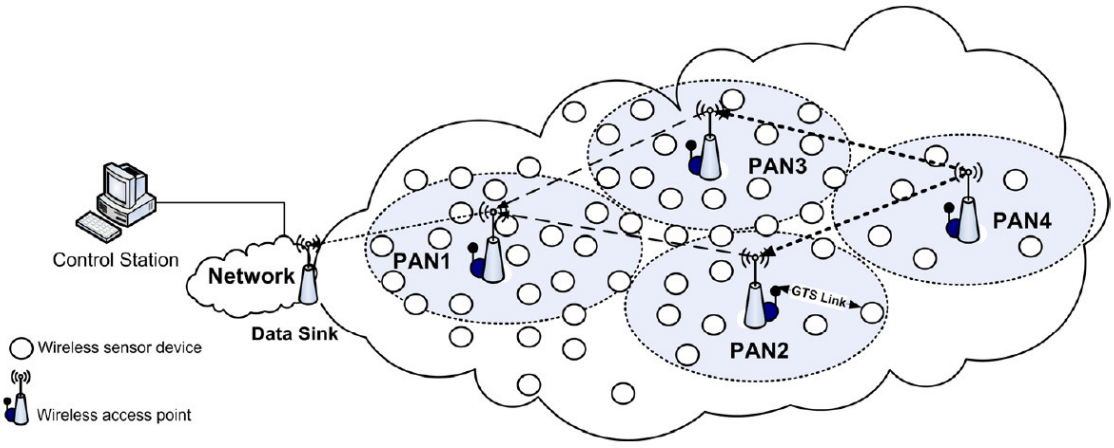


Figure 12: Art-WiSe architecture [23]

When integrating different types of networks through gateways, the gateways have to translate between the network protocols used in the different networks. Since network protocols in sensor networks are heavily influenced by the sensor network applications, in most of the cases the gateways have to run application-specific translator mechanisms as well. The suitability of having the gateway run application-specific translator mechanisms is discussed in Chapter **Error! Reference source not found.** of this paper.

8.1.2 Different access technologies, one network

This section discuss the second approach introduced above – how to deal with a network encompassing different link-layer technologies. The Internet is one such network, and there it exists a couple of very important horizontalisation points, which can be used to integrate different types of networks that may use different link-layer and networking technologies. These horizontalisation points are for example IP, TCP and http to name a few. In contrast to this, no layered architecture has been proposed for sensor networks, and these therefore tend to have a very vertical structure, with a network protocol that is very tightly coupled both upwards and downwards in the stack. Indeed, in many existing sensor network deployments it is often difficult to tell the exact difference between network and application.

We believe that in order for sensor networks to become pervasive, it is important to find and exploit horizontalisation points on which development can grow. At the same time, one must not forget that in order to save energy, control over the underlying technologies is important.

The authors of [21] designed a network and built-up a testbed in which they merge different subnetworks (consisting of nodes with different link-layer technologies) into one global network. To do this, they introduced a so called Transparency layer (**Error! Reference source not found.**), which is placed between link layer and the network layer. This way they are able to organize Mica motes (with an RFM TR1000 916Mhz Radio and a serial link connection) and PCs (with a serial link connection) into one network in order to exploit the benefit of link heterogeneity in a senor network.

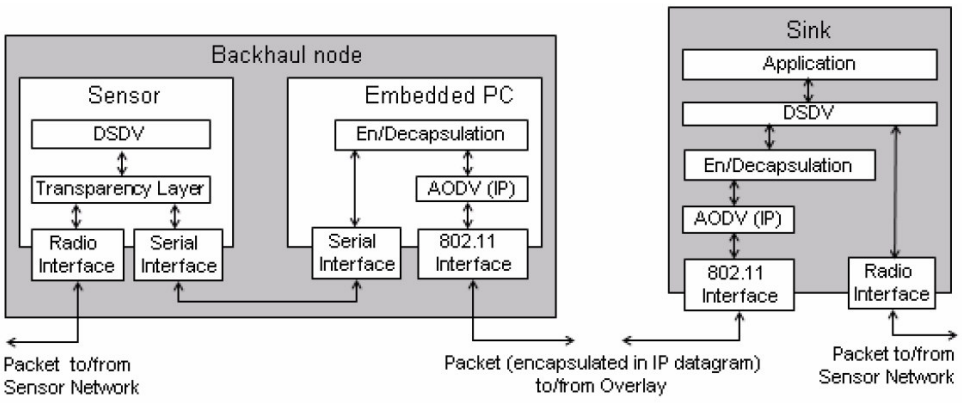


Figure 13: Hierarchical networking software infrastructure [21]

A somewhat similar solution has been proposed in [79]. In this paper the authors describe how a narrow waist could be located in between the link layer and the network layer. Their solution, which they refer to as the Sensornet Protocol (SP), is a protocol inside the sensor node between the network protocol and the link layer protocol. SP consists of two major parts – a neighbour table and a message pool. The neighbour table allows the network protocol to choose a neighbour independent of link layer technology, and the message pool allows a network protocol to send packets. Neighbour tables and message pools can be altered from both directions – both the networking protocols as well as the link layer technologies may change information in these tables. For a schematic picture of SP, see Figure 14**Error! Reference source not found.**.

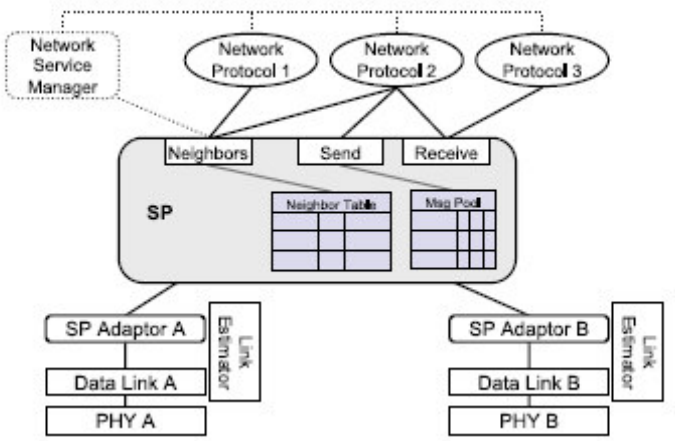


Figure 14: A schematic picture of SP (Picture taken from [79])

If one wants to utilise multiple link layers in a single network, the above two proposed solutions could be one approach. However, one thing that is missing from the solutions proposed above is that it is probably also important to allow multiple applications utilizing different network protocols to exist in the same sensor network. Different administrators might also want to use the same sensor network infrastructure using different address spaces for their networking protocols. In order to achieve this possibility, one path forward could be to let a link layer abstraction take care of the multi-hop routing. The network address is only processed on those nodes that are running the particular application that is utilizing this networking protocol. Another possibility is that the node should belong to the particular administrator using the specific address space. This implies that some kind of routing has to be performed on the link layer abstraction plane. We therefore propose to implement light forms of AODV or DYMO on the link layer abstraction. Here, the word link layer abstraction is used, since it is

possible that more data then the standard link layer header needs to be transmitted such as final destination link layer address. This kind of flexibility enabler is also proposed in for instance [80].

Just as the authors of [79] stress, it is also probable that this solution should work in environments with multiple link layer technologies. Moreover, if a networking protocol is implemented on one link layer technology, it should be easy to transfer it to another one. Therefore, a complete sensor network architecture should contain a general link layer (GLL). Both SP and ULLA [81] are different kinds of GLLs. Different aspects of these have to be considered before a decision is made on which solution is the favoured one.

Adding to this, we would like to see an architecture that includes security as an integral part of each sensor network deployment. In this scenario we see two levels of security – one is a network wide key taking care of integrity at the link layer, the other secures the payload including network address. In this solution, it should only be possible for a sensor node to read the network address if it is part of the specific application that this packet is aimed for. Here, it is assumed that the link layer is capable of handling integrity protection. The same network wide master key could be used irrespectively of link layer for the link layer integrity. Below, a typical package is depicted.

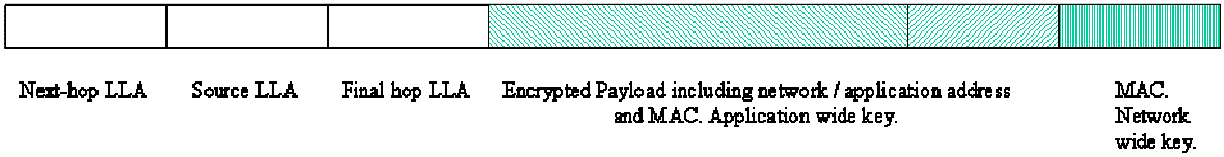


Figure 15: A typical package. Link layer addresses (LLA) are the routable address that the whole network is using. Inside the encrypted payload are the application specific addresses.

The view above is not new. Dividing security into link layer, network and application security is a very natural and traditional division. In our networks new problems however occur that are related to different users of the same applications. How to ensure privacy and to restrict certain users rights in the network are complicated by many different reasons such as messages are sent in a broadcast manner and not point-to-point, in network computation may exist that complicates the issue of deciding the origin of certain information and that many devices require lightweight security solutions. These issues are further discussed in Chap. **Error! Reference source not found.**. The specific solution that is used will of course differ depending on network topology, device composition and so forth.

8.2 Communication over sensor networks

This would be the case, if we would allow for multi-radio sensor routing nodes to communicate with each other via sensor routing nodes.

8.3 Meshed sensor network connectivity

Generally speaking, the notion of a mesh network is not very well defined. So, for our discussion it is important to introduce early our understanding of what a ***mobile mesh network (MMN)*** is in our discussion. This definition will not necessarily constrain our discussion but will act to focus our discussion towards a common idea of the network architecture relevant within the context of the sensor networks discussed here.

We define a MMN very generally as:

A network with wireless connectivity between the nodes, where nodes within the network are free to roam and where there are at least two nodes with two or more paths between them (the paths need not be orthogonal).

In considering mobile mesh networks, to start our discussion, we assume that the network architecture used is one that closely resembles a mobile ad-hoc network (MANET). The IETF work on MANETs makes the following assumptions about MANETs [82]:

1. Dynamic topologies: Nodes are free to move arbitrarily; thus, the network topology –which is typically multihop – may change randomly and rapidly at unpredictable times, and may consist of both bidirectional and unidirectional links.
2. Bandwidth-constrained, variable capacity links: Wireless links will continue to have significantly lower capacity than their hardwired counterparts. In addition, the realized throughput of wireless communications – after accounting for the effects of multiple access, fading, noise, and interference conditions, etc. – is often much less than a radio's maximum transmission rate.
3. One effect of the relatively low to moderate link capacities is that congestion is typically the norm rather than the exception, i.e. aggregate application demand will likely approach or exceed network capacity frequently. As the mobile network is often simply an extension of the fixed network infrastructure, mobile ad hoc users will demand similar services. These demands will continue to increase as multimedia computing and collaborative networking applications rise.
4. Energy-constrained operation: Some or all of the nodes in a MANET may rely on batteries or other exhaustible means for their energy. For these nodes, the most important system design criteria for optimization may be energy conservation.
5. Limited physical security: Mobile wireless networks are generally more prone to physical security threats than are fixed-cable nets. The increased possibility of eavesdropping, spoofing, and denial-of-service attacks should be carefully considered. Existing link security techniques are often applied within wireless networks to reduce security threats. As a benefit, the decentralized nature of network control in MANETs provides additional robustness against the single points of failure of more centralized approaches

However, we add additional definition to this description without loss of generality for our application scenarios, by applying the following additional assumptions:

6. Transmission: This is radio-based, naturally, and allows nodes in the network to send transmission directly to other stations within radio range. The radio link may be shared media (such as a wireless LAN) or directional point-to-point (pt2pt).
7. Communication channels: Multiple radio channels could be in use and no assumptions are made about the nature of these channels, but there may be effects of different mechanisms (shared media vs. pt2pt).
8. Data transfer mode: The underlying mode of transferring data within a channel is packet-based and connectionless at the point of transmission. That is, layer 2 (Data Link) and layer 3 (Network) are assumed to be connectionless. The mode at layer 1 could be connection oriented or connectionless, but is assumed to be asynchronous between sender and receiver. However, some form of connection oriented protocol may, of course operate on top of layer 3.
9. Traffic relay functions at nodes: As well as acting as sources and sinks of traffic, potentially, any node could also act as a relay for traffic for which it is neither the source or the sink. Some nodes may only perform relay functions and may not act as sources or sinks of traffic. Indeed, there may be one (or more) node(s) that provide(s) ingress/egress of traffic to/from the MMN.

10. Communication between nodes: the MMN may be used to provide point-to-point (pt2pt), point-to-multipoint (pt2mpt) or multipoint-to-multipoint (mpt2mpt) communication. This may be enabled at the lower layers (layers 1 and 2); or at higher layers (layer 3 and above) with or without support of the lower layers.

So, we present a very general description of a MMN.

8.3.1 Considerations for mesh scenarios

There are some key considerations that must be taken into account and a number of challenges faced for enabling mobile mesh networks.

- *Current systems, technology, systems and protocols are inadequate:* solutions for the types of scenarios being presented here do not currently exist and so it is likely the current technology, systems and protocols will not be adequate for the requirements of mobile mesh networks. However, work is in progress within the research community to address these issues.
- *Multi-layer systems and solutions:* any overall system cannot simply work in isolation in a single layer of the network stack and it is likely that real solutions, need to examine all aspects of the network architecture, from physical layer to application layer.
- *Network architecture:* routing and addressing must allow network nodes to form connectivity suitable for the mesh network operation. The current IP addressing and routing model does not cope well with multiple ingress/egress points (multi-homed mesh networks).
- *Dynamic (re-)configurability:* Nodes must be able to dynamically configure their usage of the network in the face of mobility, locations of ingress/egress nodes, traffic patterns and power consumption.
- *Congestion control and QoS:* there should be some way to control the transmission and use of radio channels within the mesh network. Some applications may cause congestion, hurting themselves and the operation of other applications.
- *Use of communication channels:* radio channel management may need to be integrated into (or at least work in close harmony with) the higher layer functions that are incorporated routing protocols and application protocols.
- *Security:* with mobile nodes entering and leaving mesh networks, there would be a need to have some sort of security architecture to control access to the radio channels (at the very least).
- *Scalability:* the scalability of our scenarios needs to be examined (number nodes in a network, traffic overhead due to routing protocols, performance of the network, etc.).

8.3.2 Routing Challenges in Meshed Topologies

The heterogeneous nature of connectivity available to the types of devices being described here, means that the meshed topologies that are formed, exhibit some key characteristics that distinguish them from existing classes of networks.

8.3.2.1 Administrative Responsibility

The meshed topologies being formed here are essentially multi-homed, with several ingress and egress connections to the wide-area (i.e., Multi-radio sensor routing nodes that have 2G/3G connectivity). Administrative responsibility may be distributed across the nodes forming the network. For example, the sensors that line the tunnel wall may be owned and controlled by local government authorities, but

the sensors that ride on vehicles are owned and controlled by the vehicle owners themselves. Thus, although the different groups of devices may interconnect, they do not necessarily represent a single Administrative Domain (AD) that is under the control of a single organisation or entity, but rather a collaborative group of such entities. However, existing mechanisms for addressing and routing are designed to operate in network environments where administrative responsibility is not distributed, and therefore may not provide the most optimal solution for interconnectivity.

8.3.2.2 Deployment and Maintenance Complexity

It is essential that for these devices, any interconnectivity solution be simple and light-weight. Therefore, the obvious choice of BGP [78] to handle distributed administrative responsibility becomes infeasible. Not only would it be too heavy-weight a solution to be deployable, but the types of low-power, low-resource devices being used here do not have the capability to support the relevant protocol and policy systems required. Even if the capabilities did exist, the large numbers of devices involved magnifies significantly any deployment and maintenance complexities.

8.3.2.3 Multi-Homing and Connectivity Sharing

Traditional ad hoc routing mechanisms have focussed on finding the single most efficient route on a source-to-destination basis, where the destination may be either inside or outside the local domain or network. This models an ad hoc network as a single AD that is either disconnected, or a direct extension of a larger infrastructure. This in turn requires that each device either discover efficient routes to a very wide set of destinations, or route towards a specific designated network gateway (representing a single point of failure).

However, the multi-homed nature of the topologies being formed here allows them to be treated as composite, multi-homed, virtual entities. In this sense, they have a high *potential* capacity to the wide-area. Unfortunately, existing mechanisms for connectivity sharing tend to concentrate on sharing a single connection between multiple machines and do not take into account the possibility of utilising multiple connections simultaneously.

There is thus an opportunity to better utilise this high potential wide-area connectivity that is available, for example by aggregating all multi-radio sensor routing node 2G/3G connections. This may be achieved by using the relatively high data rate local-area 802.11x radio links between the devices as a backplane over which traffic may be distributed across all multi-radio sensor routing nodes that have 2G/3G radios for egress from the network.

8.3.3 A Coalition-Based Connectivity Approach

The use of community-area networking techniques may provide a valuable mechanism for communication within these meshed topologies of nodes (or communities of devices) being formed here. One such technique takes a coalition-based approach whereby nodes that have proximity to each other, communicate and form a Coalition Peering Domain (CPD) [77].

Briefly, nodes that have both local-area connectivity and connectivity to the wide-area may co-operate to form the edge of the CPD. In the context of the tunnel fire scenario topology (described in Section 2), several multi-radio sensor routing nodes with wide-area 2G/3G connectivity co-operate to form the edge of a CPD. By using their often higher data rate local-area radio links as a backplane, they distribute outgoing traffic across their wide-area radio links thus utilising all the available wide-area capacity available at the edge of the CPD.

The multi-radio sensor routing nodes forming the edge of the CPD co-operate also with normal sensor routing nodes and other multi-radio sensor routing nodes that have only local-area radio links, thus allowing them to benefit from wide-area connectivity. These CPD-internal nodes need route traffic

only to the edge of the CPD. Once at the edge, traffic is distributed, as described above, across all multi-radio sensor routing nodes that have 2G/3G connectivity. **Error! Reference source not found.** illustrates how such a coalition of co-operating sensor nodes may be formed.

The formation of a coalition of sensor nodes may be undertaken through the use of geographic addressing approaches, whereby the control data and handshaking negotiation processes are geocast across a local geographic region.

In case of fire, once re-routing has been established (as illustrated in Figure 2), the distribution of traffic across the CPD edge enables the best use of the available wide-area capacity, thus allowing more information about the emergency to be transmitted quickly.

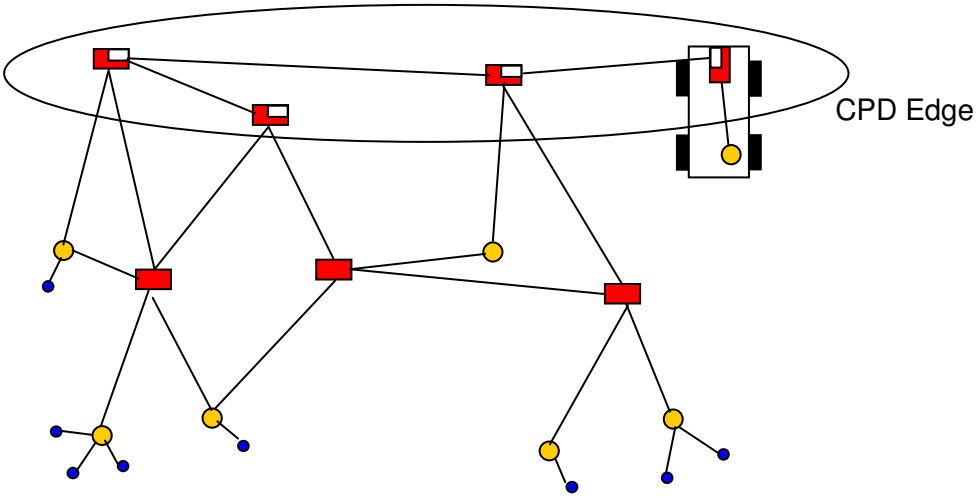


Figure 16: CPD Architecture within RUNES meshed topologies

8.4 Conclusions and Future work

The coalition-based connectivity approach lends itself well to the types of scenarios applicable within RUNES. The limitations of existing networking mechanisms to deal with the levels of multi-homing and distributed administrative responsibility involved, mean that this approach is highly desirable and beneficial within RUNES. To this end, the work described here shall be developed further not just in relation to the main tunnel fire scenario, but also within the wider context of RUNES.

For future work we identified the following topics:

- **Unicast data delivery**: how to route and deliver packets between the sensor and ad hoc network nodes. This implies using some kind of hierarchical routing mechanisms.
- **Multicast data delivery**: it should be investigated, how to send multicast messages in the sensor network and how the multicast groups are formed. It is possible to build up a multicast tree in advance, or it can be done on demand when necessary.
- **Broadcast data delivery**: it should be analysed how far a broadcast message has to be sent in the sensor network, since the ad hoc network above the sensor network can deliver the broadcast messages directly to nodes involved in the ad hoc network. This way the sensor nodes between the concerned region and the source/sink are not used for the delivering of the message.
- **Policy based routing**: a node may have multiple interfaces, therefore policies has to be used to define which packet is allowed to be sent over which interface. Policies may change according to

needs of the applications and users. Since a network can be used by different applications and users at the same time, different policy profiles have to be able to co-exist on the same node.

- **Mobility**: if needed by an application it should be detected if a node changes its location and the routing has to be changed accordingly.
- **Attach/detach of nodes**: how will new nodes attach to the system and take over the responsibilities of destroyed nodes in the tunnel's infrastructure. Basically there are two possibilities for that, either they simple answer to incoming queries or they inform the network about their existence before they answer queries.
- **Network management**: how to distinguish between different sensor network regions in terms of management. For example, the sensors in the firemen PANs should be merged into the tunnel's infrastructure, however they still should be controlled by firemen PAN controller.

9 Security in RUNES sensor networks

Wireless sensor networks (WSN) are networks consisting of small, lightweight devices with limited computing power, memory capabilities and energy resources. Even so, the number of applications and scenarios where sensor networks can be used are huge. Application possibilities stretch from monitoring large geographical areas like farm fields, forests and coasts, to small Body Area Networks. The purpose of the first group could be to assist the farmer in using the right amount of water and fertilizer, whereas the second group can for example enable a physician to give a patient an early discharge. Both the above applications are usually considered to have very dedicated networks, where the network, the user and the application is very tightly tied to each other and no dynamics can be allowed. This is however not always the case. We will in this paper discuss two scenarios where there is a need for more complex sensor networks solutions. Since sensor nodes are tiny devices with limited operating systems, it is also important to raise the question about what is reasonable to achieve in the near future. Moreover, different architectural possibilities for more complex wireless sensor networks are proposed.

9.1 Preliminaries

We will in this section introduce a terminology that we will use in the rest of the paper.

Sensor node: A lightweight device with wireless radio and limited resources when it comes to for example energy, memory and computational capacity. The sensor node may have sensors and/or actuators and it is the main building block in a wireless sensor network.

Cluster head: A somewhat more capable sensor node that can handle more complicated tasks as for example asymmetric cryptography.

Wireless sensor network (WSN): A network consisting of sensor and routing sensor nodes, communicating using the same technology and possibly also distinguished by sharing a cryptographic secret.

Gateway: A node functioning as the bridge between the sensor network and the (IP based) WAN. The gateway is oftentimes much more resourceful than the sensor and sensor routing nodes.

User: A legal entity that uses an application in the network.

Application: A set of sensor nodes that interacts in order to perform a specific task. In particular, there may be more than one application running in the same WSN, and these do no have to be related to each other in any way more than that they utilize the same communication infrastructure.

It is important to note that we consider a sensor node as consisting of three parts: a networking part, a computing part, and a sensor part. This means that a sensor node may merely relay messages under some circumstances, and may be an end-point under other. If a sensor node just relays a package, that particular sensor node should not be able to modify or view the content of that package. On the other hand, it is reasonable to say that a sensor node that needs to view and/or modify the content of the package is considered to be an end-point.

9.2 Scenarios

The general scenario that is considered in this deliverable, the tunnel scenario, is not considered in depth in the first sections in this chapter. The reason for this is that it is not showing all the difficulties that can show up regarding different managers, even though problems regarding different users are abundant. We therefore chose to describe two other scenarios as a complement to the tunnel scenario.

In section 9.6, the tunnel scenario is however an excellent fit.

9.2.1 The Skyscraper

Consider a large skyscraper. In a skyscraper there are apartments, offices, restaurants and so forth. All these premises belong to different owners. At the same time there are many different kinds of sensor networks installed throughout the building: burglar alarms, thermometers, fire alarms, electrical power meters, water meters, sensors keeping track of open doors, which lamps that are turned on, when plants need to be watered, the tension in the walls, and also actuators like sprinklers, lamps and door closers.

A traditional architecture would promote a solution where sensors and actuators are connected into dedicated networks depending on their usage and/or location. For example, all the sensors that are installed in order to measure the humidity in the plants make up a network on their own. This is what we believe is not the case – we believe that these dedicated networks will be connected to all other sensor networks in the building.

The reason for interconnection is in this case mainly coverage – since sensor networks are lightweight and have a limited communication range, we believe that they would benefit from using each other as relay devices even though there is no direct communication between specific devices. A related reason is resistance against physical attacks – the more sensor nodes that are interconnected; the more resistant is the network to jamming or broken sensors. For instance, if a specific sensor node is broken, the surviving sensors may route around that lost node.

In other situations, the sole reason for connecting networks belonging to different owners, might be that hey are interested in the same sensor values. Applications and users are overlapping each other in a way that is not always unambiguous. As an example, consider the temperature sensors in an apartment. Both the apartment and the skyscraper owner might be interested in this information. The caretaker in the skyscraper is possibly also interested in the temperature in all other rooms of the building. If the sensor network had been implemented as small, dedicated networks in every apartment, the possibility for an authorized user to do a wider search in the sensor network would be lost, and he would instead be forced to collect the information via all networks one by one.

Combining sensor networks like this into a larger network can however not be done without solving many problems. For instance, privacy reasons may dictate that the data in the above scenario should not be transmitted to anyone else in the network but the users with the appropriate authorizations.

9.2.2 The Cargo Ship

Another scenario where there are reasons to connect sensor networks concerns shipping. This scenario has recently been used to illustrate a multi-application environment in **Error! Reference source not found.**. Let us take a large cargo ship with containers as an example. Each container contains an RFID reader and a number of sensors measuring the temperature, humidity, etc. The goods itself is also equipped with RFID tags and sensors. The RFID tags help to keep track of where a particular goods is

located (in which container on which ship and so forth.) The sensors monitor that temperatures and other parameters are kept on the right level. The sensor network inside the container is connected to the outside world via a satellite connection. A satellite connection is a somewhat unreliable and expensive connection, which causes at least three effects. First, it implies that at least some level of surveillance on the goods should be local in nature, i.e., the ship owners may need to require some data from inside the container. Second, the satellite connection may go through a centralized point on the ship, in order to not force containers to be equipped with their own satellite connection. Third, the communication between the sensor network and the company owning the goods must be tolerant to delays and disruptions.

Both the first two effects require that sensor networks with different owners and applications are connected to some degree. It is for instance not wise to let all communication go via the home company, which in turn replies back to the ship if something has happened. It is much more effective to have this surveillance on the boat itself, and it is easy to believe that they use the same sensors as the home company does. The motivation for using one satellite connection is easy to see, but it also puts extra requirement on the solution. There might for instance be so that the home company does not trust the shipping company with not changing their temperature data (and their might even be an incentive for them to change it) or to see their RFID data, so that all data that is received by the home company is both confidentiality and integrity protected.

The third requirement obviously points towards the necessity of using intermediate storage somewhere on the line between the sensor nodes and the home company.

9.2.3 Trust models

If one try to distil different trust models from the above two scenarios, one have the three different situations depicted in **Error! Reference source not found.**.The situation in a) corresponds to the traditional view of sensor networks where all devices are trusted and the user has access to all sensor data. For instance the caretaker in the Skyscraper scenario corresponds to this user scenario. Scenario b) concerns a scenario where only some of the sensor nodes are trusted or where the user only has access to data from some of the sensors. But, the user trusts the gateway to the WAN network, and this can hence be a security endpoint to the outside. And finally, in c) the situation is similar to b), but the user does not trust the gateway to alter or read data. The security endpoint must hence lie somewhere inside the sensor network. In the cargo ship scenario, scenarios b) and c) could for example correspond to the boat and cargo owner, respectively.

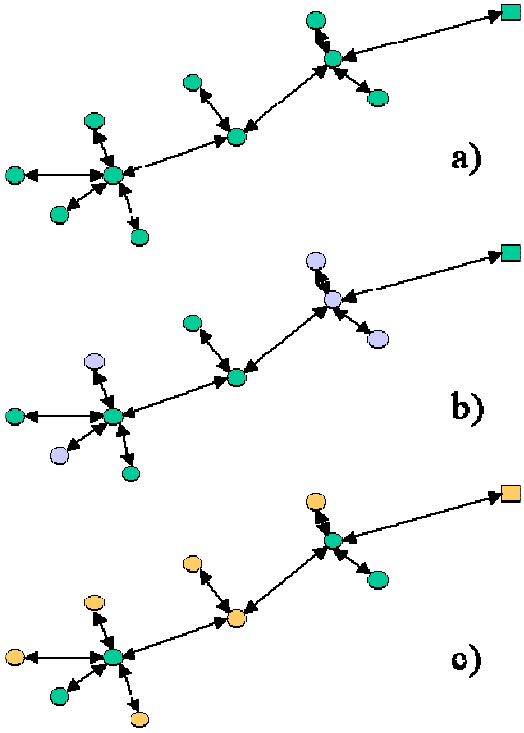


Figure 17: Three different situations in the same network. In a) all devices in the network is trusted by the user in the network. In b) there are certain sensor nodes that a specific user is not allowed to use. In c), the user does not trust the gateway. Green nodes are considered to be trusted.

Another way to view the problem is from a user / physical location / application perspective. Consider an application as a collection of sensor nodes performing a specific task such as temperature measuring. Now, different users may exploit different parts of the physical world depending on what data they try to reach (such as in for example the Skyscraper scenario). We then get a view of the sensor network where specific users have access to sensor data depending on at least two dimensions – physical location and application. Even more dimensions can be considered; some users might for example have the right to set certain parameters in specific applications at specific physical locations. Network and node management can also be considered as different rights. A system with two access dimensions may be depicted as in **Error! Reference source not found.**.

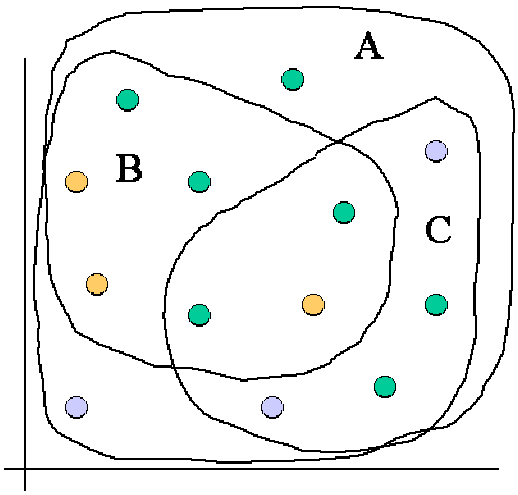


Figure 18: User A has access right to the green and the blue application, user B to the green and yellow, and user C to only the green. All are restricted to their respective areas. No sensor is in this schematic picture assigned to more than one application, which could very well be the case in a general model.

It is noteworthy to compare the problems we are discussing with one of the problems of early OS security: how to construct a system that allows more than one user to utilize the same computing platform in a secure way **Error! Reference source not found.**. In this setting, the computing platform is however made up of many small, independent devices instead of a single large, monolithic structure. The problem is larger in systems like the one we are considering, since we cannot assume a secure communication infrastructure, as they did in **Error! Reference source not found.** and, as in Cargo ship scenario, no single trusted central entity.

9.3 Network Topology

Depending on the network topology we may have different problems to solve. There are three topologies that we need to consider:

- Full mesh
- Tree-based
- Hierarchical cluster based

Examples of these topologies are depicted in **Error! Reference source not found.**.

In a full mesh topology, all nodes in radio distance from each other may communicate. In this topology, all nodes are routing sensor nodes.

In the tree based topology, there are leaves that can only communicate with a specific sensor node. These are sensor nodes. Then, there are nodes that can communicate with multiple nodes. These nodes are routing nodes. Communication in a tree based topology is always in a certain direction – up or down.

In the hierarchical-cluster based topology the sensor nodes are split into different groups called clusters. Internally in these groups there is a somewhat more capable entity that functions as a cluster controller (or *cluster head*). This entity communicates with other cluster heads, but not with sensor nodes in other groups. The topology inside the cluster can either be full mesh, tree-based or even hierarchical cluster based again.

The issue of topologies is interesting because the possible choices of security architectures are dependent on the network topology. Or, one can view it from the other direction, the network topology could be chosen from the security requirements.

Different usages, i.e., different applications and users, management of sensors and network and so forth, may also have different topologies in a single network. This is possible if all communication can be viewed as part of an overlay network, separated from the others.

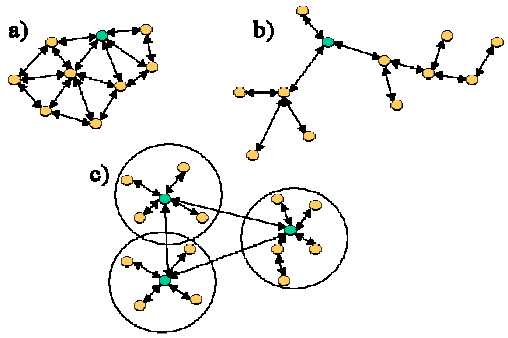


Figure 19: Three different network topologies. a) Full mesh, b) Tree-based, and c) Hierarchical-cluster based. A green node is assumed to be more powerful than a yellow.

9.4 Architectures

The architecture possibilities for sensor networks that we will consider have two dimensions – the location and duties of policy enforcement points, and the choice of WAN connection method.

Often, it is assumed that the gateway not only functions as the bridge from the sensor network to the WAN (IP-based) network on a pure technological level, but also that it acts as a security endpoint towards the WAN. With this is meant that the sensor network is controlled by the gateway, and that the gateway not only enforces a user's network access, but also his data access. The gateway is hence the sole entity that makes access decisions based on user information. In this case, with the definition of security end-point introduced above, it means that the gateway must be fully trusted by all users utilizing the sensor network. The Cargo Ship scenario shows that this is not always the case – in this scenario the user trust the gateway for network access but not for data access. Another option could hence be to let the gateway purely handle network access control. If the gateway is alleviated from data access control, this has thus to be enforced inside the network. Another drawback with this solution is that sensor networks often utilizes in-network computation such as data aggregation and modification, and it is therefore difficult to know where data has originated.

Where should data access be enforced, then? Well, in the end, enforcement has to be implemented even in the smallest sensor nodes - but this enforcement can be very simple and based on proving knowledge of a secret key. For instance, one or more powerful nodes might control a simple sensor node. The more powerful nodes then need to make the decision and enforce which applications and users that have access to the less powerful nodes. This tactics of letting the more powerful nodes do the security decisions correspond to a hierarchical network topology. But, in doing so, we also restrict the possible network topologies. If all policy decisions have to go via a specific node, all traffic also has to flow via this node.

If we would like to create architecture that allows for a very generic deployment of network topologies, like for instance fully meshed topologies, then we need even more distributed security decision points. This implies that we need to have user authentication and authorization in each sensor node. In order to achieve the most versatile and flexible user-authentication, we must thus enforce it in every sensor node. One way of achieving this is to establish a common secret among the sensor nodes that runs a certain application and a specific user is allowed to use. In essence, we establish an overlay for a specific application and a specific user such that this user can use the allowed sensors independent of the underlying infrastructure. Here, the establishment of keys is crucial and there exist many key management schemes for sensor networks in the literature. These include for instance LEAP **Error! Reference source not found.** and PIKE **Error! Reference source not found.**. The problem with all of these schemes is that they focus on establishing a key throughout the whole sensor network. We have another goal. We look for a solution where we can instead confine the key establishment to certain users and applications to create the overlay.

There have been a number of proposals regarding how the WAN should connect to the sensor network. These can basically be divided into the following three types:

- Application resolution.
- Using IP end-to-end.**Error! Reference source not found.Error! Reference source not found.**
- Tunneling sensor network protocol over Internet.**Error! Reference source not found.**

Which one of these you choose of course impact if the gateway can be considered part of the application/user overlay or not, i.e., if it enforces data access or not. If we use application resolution in the gateway, then we are forced to trust that entity. As mentioned above, that may not always be the case. Using IP in an end-to-end fashion is more appealing in that respect, since it is easy to move the user authentication closer to the actual data sources. In this setting we however will have problems regarding multicast and the heavyweight nature of the IP protocol, especially from a security point of view (even if shared key version of IPsec exist, and a similar version is being worked on for TLS). The third alternative is to tunnel the sensor network protocol to the user. This requires the user to be aware of the addressing used in the sensor network. This may of course be difficult, but as discussed previously in for instance **Error! Reference source not found.**, application, addressing and routing are generally very tightly bound to each other in a sensor network. If this indeed is the case, this does not really constitute a problem for the application. It still has to be there. This approach makes it easier to push user authentication and authorization for data access.

9.5 Distributed Management

Independent of the topology or key management schemes that is used, we need to figure out how the network is supposed to be managed. As proposed before, management can be considered to be an application. However, in the envisioned sensor network, where different users have access to different parts of the network, it is natural to also have a distributed management. For instance, an apartment owner may want to manage the part of the network which is inside his apartment. We can therefore look upon the management of the network as in **Error! Reference source not found.**.

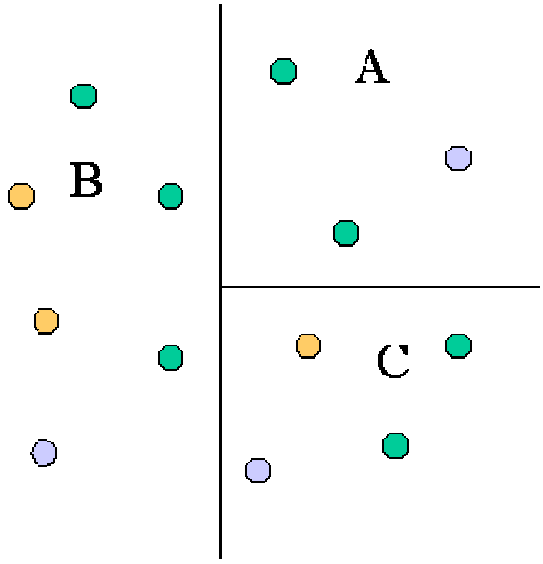


Figure 20: The picture shows a sensor network with distributed management. Three entities, A, B, and C, are managing different parts of the network.

It is a great challenge how to come up with an architecture that allows for distributed management without the managers being able to compromise for each other if only symmetric primitives are used.

9.6 Group Key Management

We consider a network in which computing nodes cooperate towards a given application according to the *group communication* model. In this model, a node becomes a new member of the group by explicitly joining it. As a member of the group, the node may broadcast messages to the other members. Later on, the node may voluntarily leave the group, or, if compromised, may be forced to. After that event, the node cannot send messages to, or receive messages from, the group. The group can in this case be any kind of group; it can be either a whole network or an overlay based on application or user.

In order to protect group communication from both a passive and an active external adversary, group members share a *group-key* they use to encrypt messages within the group so that anyone that is not part of the group can neither access nor inject/modify messages. We require that when a node joins the group, it must not be able to decipher previous messages encrypted with an old key even though it has recorded them (*backward security*). Furthermore, when a node leaves, or is forced to leave, the group, the node must be prevented from accessing the group communication (*forward security*).

In this model, we provide forward and backward security via a *reactive rekeying protocol*, in which the group-key is changed and distributed whenever a node joins or leaves the network. The rekeying protocol is conceived for large, highly dynamic groups and ensures scalability in that it limits both communication and computation overhead during rekeying [106][107]. The protocol is based on hash functions and symmetric ciphers and levers on two basic mechanisms: *key-chains*, an authentication mechanism based on the Lamport's one-time passwords [108]; and a *logical key hierarchy* (LKH), a technique for secure and scalable group rekeying [115][116].

Intuitively, LKH allows us to achieve a rekeying protocol which requires a number of messages that is a logarithmic function of the network size, and also logarithmic memory. This is an improvement with respect to naive schemes that accomplish either linear complexity in transmission overhead and constant storage, or constant transmission overhead but exponential in memory. Another problem with group communication concerns source origin authentication (SOA). This problem can be solved by introducing some kind of asymmetry. This asymmetry can either be based on space, time, or an asymmetric function. Here we will only discuss the two latter, but a reference to space-based SOA can be found in for example [117]. μTesla, for example, achieves SOA but at the cost of some delay in time. μTesla can hence not be used to authenticate real-time transmissions, and with respect to key-management a vulnerability window will occur due to a delayed key authentication. SOA based on an asymmetric function uses regular digital signatures. These have the drawback that they are computationally expensive and require large keys, which in turn make them unsuitable for lightweight environments. It must be said that we are not the first one that use LKH for key distribution in a large scale distributed system. Relevant applications can be found in [111][114][115]. However, these systems use digital signatures based on public key, e.g., RSA [113], to ensure the authenticity of new keys. In contrast, we achieve such a proof of authenticity by means of key-chains, a much more lightweight mechanism that uses only hash functions. By this mechanism, a sensor node just needs to use a hash function to verify the key authenticity. Furthermore, key-chains allow us to reduce the size of rekeying messages with respect to LKH as it is not necessary to append the digital signature on the key anymore.

In LEAP every node shares a key with its neighbours **Error! Reference source not found.**. Therefore, if the neighbourhood is composed of D nodes, on average, then every node has to store $D + 1$ keys. When a node leaves the group (e.g., it is compromised), every neighbour of that node has to renew its key and transmit it to its own neighbours in its turn. This action is performed by unicasting

the new key to every neighbour, encrypted by means of the corresponding link-layer key. It follows that rekeying requires D^2 messages. It follows that the overhead of the rekeying protocol is a quadratic function of the neighbourhood size. However, upon receiving a data message, a node has to decrypt it and re-encrypt it with the key the node shares with its neighbours before broadcasting the message to them. For example, assume that node *A* sends a message encrypted key K_{na} that it shares with its neighbourhood. If node *B* is a neighbour of node *A*'s, then *B* has to decrypt the message with K_{na} and re-encrypt it with K_{nb} before broadcasting it. In contrast, in our model, sensor nodes share a single group-key for encrypting and decrypting messages and therefore they do not need to decrypt/encrypt every message. It follow that LEAP's rekeying protocol is less expensive in terms of number of messages than ours—given the neighbourhood size is smaller than the network size—but it is certainly more computational expensive during the normal communication phase.

The *Key Distribution Service* is the architectural component that is responsible to perform such a rekeying task upon every group membership change. In a centralized implementation of the Key Distribution Service, a single entity called the *Group Controller Key Server* (*GCKS*) is employed for renewing the group-key whenever the group membership changes in order to guarantee backward and forward security. We assume that *GCKS* is a gateway, a powerful computing node with plenty of power, computing, memory, and communication resources. Furthermore, we assume that *GCKS* is physically protected or is equipped with tamper-resistant hardware, so that we can rule out that *GCKS* can be compromised.

9.6.1 Key-Chain: A mechanism for authentication

A key-chain is a set of symmetric keys, linked to one another by means of a one-way hash function. Each key in the key-chain is the image of the next key under the one-way function and every group member can locally verify every received key from the preceding ones [106].

Let *H* be a one-way hash function (OWHF), that is, a hash function with the following two additional properties: (a) given an output *y* for which the corresponding input is not known, it is difficult to find an image m' such that $H(m') = y$ (*preimage resistance*); and (b) given an input *m* it is computationally infeasible to find a second preimage m' such that $H(m') = H(m)$ (*2nd-preimage resistance*). Examples of functions considered to be OWHFs are MD5 [112] and SHA-1 [109], even though weaknesses have lately been found in both.

Given a OWHF *H*, *GCKS* chooses a secret *s* and builds a key-chain composed of $N+1$ elements as follows: s, $H(s)$, $H^2(s)$, .. , $H^N(s)$, where $H^i(s)$ corresponds to applying i times *H* on *s* and $H^0(s) = s$. We denote by $K_i = H^{N-i}(s)$ the *i*-th key of the key-chain and call $K_0 = H^N(s)$ the *chain head*. Properties (a)–(c) of OWHF guarantee that, with the knowledge of the key K_j, anybody can compute all the previous keys K_i, $0 \le i \le j$, but cannot compute any of the later keys for $j + 1 \le i \le N-1$.

GCKS uses a key-chain for authentication purposes as follows. Assume that *GCKS* transfers a given key K_i of the chain to a given node *A* through a predefined authentic channel. If, later, *GCKS* transmits K_j, $i < j \le N$, to *A* over an insecure channel, then *A* can verify the authenticity of K_j, i.e., it actually comes from *GCKS*, by applying $j - i$ times *H* to K_j and checking that the result is equal to K_i. That is, $K_i = H^{j-i}(K_j)$. It must be noticed that, if *GCKS* initially transfers the chain head K_0 to *A*, then *A* can authenticate all keys in the key-chain.

9.6.2 The rekeying protocol

We assume that every node shares a secret, symmetric key with *GCKS* that we call the *private key* of the node. We denote by K_A the private key of a node *A*. We assume that every node is initialized with its own private key before being deployed.

9.6.2.1 Forward security

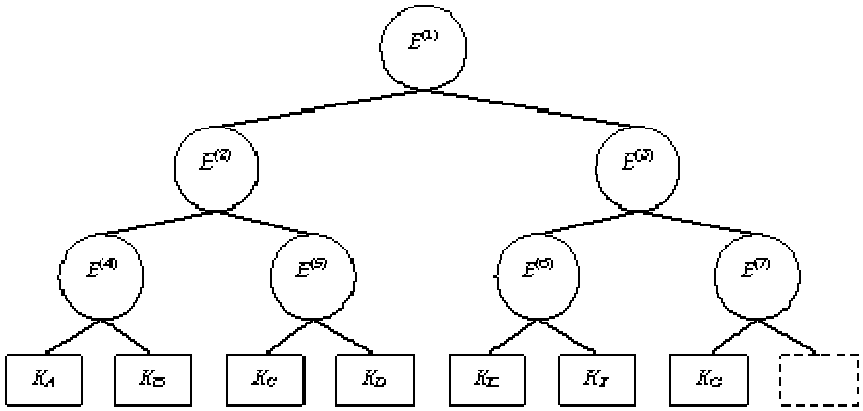


Figure 21: The eviction tree.

In order to preserve the forward secrecy, *GCKS* maintains a hierarchical structure of cryptographic symmetric keys, called *eviction-tree*. The internal nodes are numbered in Breadth-First Search (BFS) fashion where the root is numbered 1. Tree nodes are associated with encryption keys as follows. Every leaf is associated with the private key of a group member whereas every internal node is associated with a key-chain and a key pointer. At any given moment, the key pointer specifies the current key of the chain, that is, the key was used last. When *GCKS* has to determine the next current key of an internal node, *GCKS* has simply to increment the key pointer by one. In the following we denote by $E^{(i)}$ the current key of the key-chain associated to the internal node *i*.

Every group member stores a *key set* that contains the current keys of every internal belonging to the path from the root to the leaf associated to the node. It follows that the current key of the root is stored in the key set of all group members and therefore can be used as the group-key. For an *m*-ary balanced tree, every node has to reserve memory for the storage of $\log_m(n)$ keys, where *n* is the number of nodes in the network. With reference to Figure 21, let us consider the node D, for example. The path from K_D to the root contains nodes 1, 2 and 5. Therefore, the node's key set contains keys $E^{(1)}$, $E^{(2)}$, and $E^{(5)}$.

When a member leaves, or is forced to leave, the group, all the keys it holds become *compromised*. In order to preserve the forward secrecy, *GCKS* has to renew and redistribute all the keys belonging to the key set of the leaving node. Initially, for each internal node in the path from the root to the leaf of the leaving node, *GCKS* chooses the new current key. Then, *GCKS* removes the leaf associated to the leaving node from the eviction-tree. Finally, *GCKS* generates rekeying messages to distribute the new current keys. For each new current key, and for each child of the key, *GCKS* generates a rekeying message containing the new current key encrypted with the key of the child. As the private key held by the leaving member was not used to encrypt any new key, and all its known current keys were changed, it is no longer able to access the group messages.

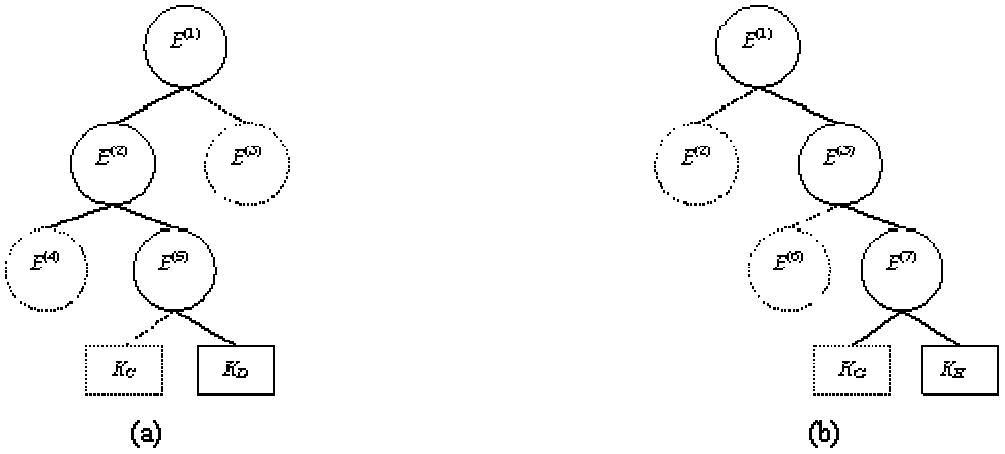


Figure 22: Managing the group membership: (a) leaving of node *D*; (b) joining of node *H*.

With reference to Figure 22a, let us assume that the node *D* leaves the group. It follows that keys $E^{(1)}$, $E^{(2)}$ and $E^{(5)}$ become compromised and *GCKS* has to substitute each of them with the next one in the respective key-chain. This means that, if we denote by $E_*^{(1)}$, $E_*^{(2)}$ and $E_*^{(5)}$ the new keys, where $E^{(i)} = H(E_*^{(i)})$, $i = 1, 2, 5$, then *GCKS* generates the following rekeying messages containing the new keys encrypted with their respective children's current keys:

M1. $GCKS \rightarrow C$: $enc\ (K_C, E_*^{(5)})$
M2. $GCKS \rightarrow C$: $enc\ (E_*^{(5)}, E_*^{(2)})$
M3. $GCKS \rightarrow \{A, B\}$: $enc\ (E^{(4)}, E_*^{(2)})$
M4. $GCKS \rightarrow \{A, B, C\}$: $enc\ (E_*^{(2)}, E_*^{(1)})$
M5. $GCKS \rightarrow \{E, F, G\}$: $enc\ (E^{(3)}, E_*^{(1)})$

where $enc\ (X, Y)$ denotes the encryption of quantity Y with key X.

Upon receiving the re-keying messages, the group members need to decrypt them with appropriate keys in order to get the new keys. Then, the group members can immediately verify the authenticity of the new keys by applying the one-way hash function. With reference to Figure 22a, let us consider the node *C*, for example. Upon receiving messages M1, M2, and M4, the node performs the following actions. Initially, the node decrypts M1 with its private key K_C and checks that $E_*^{(5)} = H(E_*^{(5)})$. Then, the node decrypts M2 with $E_*^{(5)}$ and checks that $E^{(2)} = H(E_*^{(2)})$. Finally, the node decrypts M4 with $E_*^{(2)}$ and checks that $E^{(1)} = H(E_*^{(1)})$.

9.6.2.2 Backward security

Although the eviction-tree gives support to forward security, it is not suitable to preserve the backward secrecy. Actually, when a new node joins the group, it is associated with a leaf of the eviction-tree and receives the current keys of the internal nodes belonging to the path from the leaf to the root. However, as soon as the node comes into possession of these keys, it also becomes able to access information items encrypted prior to its joining by simply calculating the necessary decryption keys with the one-way hash function.

With reference to Figure 22b, let us suppose that the node *H* joins the group. *GCKS* inserts K_H as a child of an internal node (e.g., 7) and unicasts *H* a rekeying message containing the keys $E_*^{(1)}$, $E_*^{(3)}$, and $E_*^{(7)}$ encrypted with *H*'s private key K_H. As soon as it receives these keys, *H* is able to easily compute the keys that precede them in the respective key-chains. For instance, *H* can compute the previous group-key $E^{(1)}$ as $E^{(1)} = H(E_*^{(1)})$. Notice that renewing keys associated with the internal nodes with the next ones does not help. For instance, if *GCKS* renews $E_*^{(1)}$ with $E_{**}^{(1)}$, the next key in the

chain, the joining node H can still recompute both $E^{(1)}$ and $E_*^{(1)}$ by simply applying the hash function: $E_*^{(1)} = H(E_{**}^{(1)})$ and $E^{(1)} = H(H(E_{**}^{(1)}))$.

In order to solve this problem we use an additional symmetric key shared by all current members of the group that we call the *join key*, J. The join key is securely renewed whenever a new node joins the group so that it cannot obtain the previous one. The group-key can now be obtained by passing J and the current value of $E^{(1)}$ through an hash function. Backward security is now guaranteed because the new node does not know the previous join keys, and thus is not able to recover the corresponding previous group-keys.

Instead of generating a fresh join key and unicasting it to members already in the group, *GCKS* let each such a member locally calculate the new join key so reducing communication costs and saving energy. This is accomplished by simply applying the OWHF to the current join key. Consequently, *GCKS* needs to unicast the new join key only to the joining node, encrypting the message with the node's private key. With reference to the previous example, *GCKS* sends the node H the following encrypted message $enc\ (K_H, J \| E^{(7)} \| E^{(3)} \| E_*^{(1)})$, where $\|$ denotes the concatenation operator.

Even though members can compute the new join key locally, they must be informed when it is time to do so. Therefore, *GCKS* needs to broadcast the command of renewing the join key to all nodes that are already in the group. This is a security-critical information and must be authenticated so that every group member can become sure that the command has been issued by *GCKS*.

Once again, the authentication of the message conveying the command of renewing the join key is based on the key-chain technique. In short, *GCKS* initially generates a *command key-chain* and distributes the chain head C_0 to the members of the group through a predefined authentic channel. Later, whenever a new member joins the group, *GCKS* sends the current command key encrypted with the member's private key. Notice that this encryption is not for secrecy but to provide a proof of command key authenticity to the joining node.

With reference to the above example, let us suppose that the command key is C_i when node H is about to join. *GCKS* broadcasts a message containing C_{i+1} to command the nodes already in the group to renew the join key. Upon receiving this message, these nodes check that $C_i = H(C_{i+1})$ and renew the join key. Contextually, *GCKS* sends C_{i+1} to H encrypted with K_H, i.e., $enc\ (K_H, C_{i+1} \| J \| E^{(7)} \| E^{(3)} \| E_*^{(1)})$.

It follows that this approach reduces the number of messages necessary to distribute a new group-key after a new member joins the group. In fact *GCKS* needs only to transmit two messages: a unicast message to the new member and a broadcast message to the members already in the group.

9.7 Conclusions and Future work

The traditional model of a sensor network as a very dedicated solution, with a collapsed stack and no internal user authentication / authorization, must be changed. In order to allow for more complex use cases with multiple users and applications, each utilizing their own addressing structure, a more layered approach has to be taken. Depending on the topology of the sensor network, different solutions can be conceived. In a hierarchical topology, a more straightforward approach can be taken and more standardized ways can be used in order to create a finer user authentication / authorization. However, also here, new protocols might be needed that can handle for instance multicasting of requests. The relative simplicity of this approach comes at the cost of having a more restrictive topology.

If a flat mesh topology is going to be used, we probably need to enforce authentication and authorization all the way down to the node level. On top of this, the authentication can not be done using asymmetric methods. Therefore, with this approach a completely new method has to be invented

in order to achieve distributed policy enforcement points in the sensor network. RUNES will in the next year work in both these directions.

Another choice that has to be made is how the sensor network is connected to the Internet. Which solution that is chosen will impact the possible trust models. The method that gives most freedom to the application developers to position policy enforcement points at the most places, is tunnelling of sensor network protocol over Internet.

Distributed management is also a difficult issue where research will be carried out during the next year. This is a difficult subject, but some thoughts will be developed.

The RUNES project has developed a new group keying scheme that facilitates source origin authentication for the key updates, which is a key issue in many security schemes for sensor networks.

10 Mobility of RUNES gateways

There are scenarios like automated homes or more generally buildings, in which the gateway between the sensor / actuator network and the backbone will be a fixed one, that is, geographically static. This could be for example the DSL router in a home, which additionally provides access to a sensor / actuator network via a ZigBee interface.

However, many scenarios have a strong requirement for gateway mobility. One of the scenarios RUNES is focusing on, the fire in the road tunnel scenario, exactly belongs to the group with this kind of mobile gateway requirements. For example a sensor network could consist here of sensors attached to fire-fighters measuring their body and health conditions as well as environmental sensors measuring smoke and temperatures. All these sensors could be connected to a headquarter via an edge gateway placed on a fire engine, which will connect the sensor and actuator networks to the WAN, using depending on their availability and suitability different communication technologies , such as fixed lines on-site, WLAN, cellular networks, or even satellite links.

Obviously one important property of such a gateway is the efficient support of the dynamism of these kind of scenarios. This encompasses:

- the geographic mobility of the gateway, which could be required to roam between different locations of an emergency scenario, and
- the changes of the communication technology to the backbone due to availability or reasons of suitability.

The former mobility scenario, that is, the roaming of the gateway between different geographic locations, is often dealt with by the appropriate subnet technology used. For example cellular networks allow a seamless roaming, and also for WLANs the roaming between different access points can be handled within the link layer. However, in case the roaming occurs between different communication technologies, like from a WLAN connection to a cellular connection, usually the IP address of the gateway will need to change. This “IP mobility” cannot be solved in a transparent way for the applications by link layer mechanisms; it has to be dealt with by the network layer.

The Internet Engineering Task Force (IETF) is working on solutions for supporting this “IP mobility”. Three coarse categories of this mobility support mechanisms, namely “host mobility”, “network mobility” and “mobile ad-hoc networks” (MANETs) have been coarsely described in deliverable D4.1. Which of those categories will be suitable to for supporting the mobility of gateway depends also on the way the sensor network is built:

- If the sensor / actuator nodes are not IP enabled themselves, an application specific or sensor / actuator network specific transport mechanism will be used for exchanging information between sensor / actuator nodes and the gateway. The gateway will then encapsulate this information into IP packets for further transmission into the backbone. Consequently the only IP enabled node in this kind of scenario will be the gateway itself, more precisely, its interface to the backbone. As in this case only a single “IP address” is roaming, mechanisms for “host mobility” will be most suitable for supporting such a scenario.

  A similar scenario from a gateway mobility point of view are sensor / actuator networks, which use IP addressing on their nodes, but don’t expose these IP addresses to the WAN. An example would be using private IP addressing within the sensor / actuator network, and map these by means of Network Address Translation (NAT) or application proxies to global IP addresses for WAN transmission.

- If the sensor / actuator nodes themselves are also using global IP addresses, the sensor / actuator network could be seen as a single IP subnet. Consequently a complete IP network will need to receive mobility support if the gateway changes its connection to the backbone. As in this case a complete "IP network" is roaming, mechanisms for "network mobility" will be most suitable for supporting such a scenario.

10.1 Mobility of gateways connecting non-IP enabled sensor / actuator networks

The following Figure 23 illustrates a scenario, in which a non IP enabled sensor / actuator network will be connected via a mobile gateway to different access networks to an IP-based backbone network. Also connected to the IP backbone is a location hosting applications server, which e.g. require access to the data collected in the sensor network, or which issue control commands to the actuators.

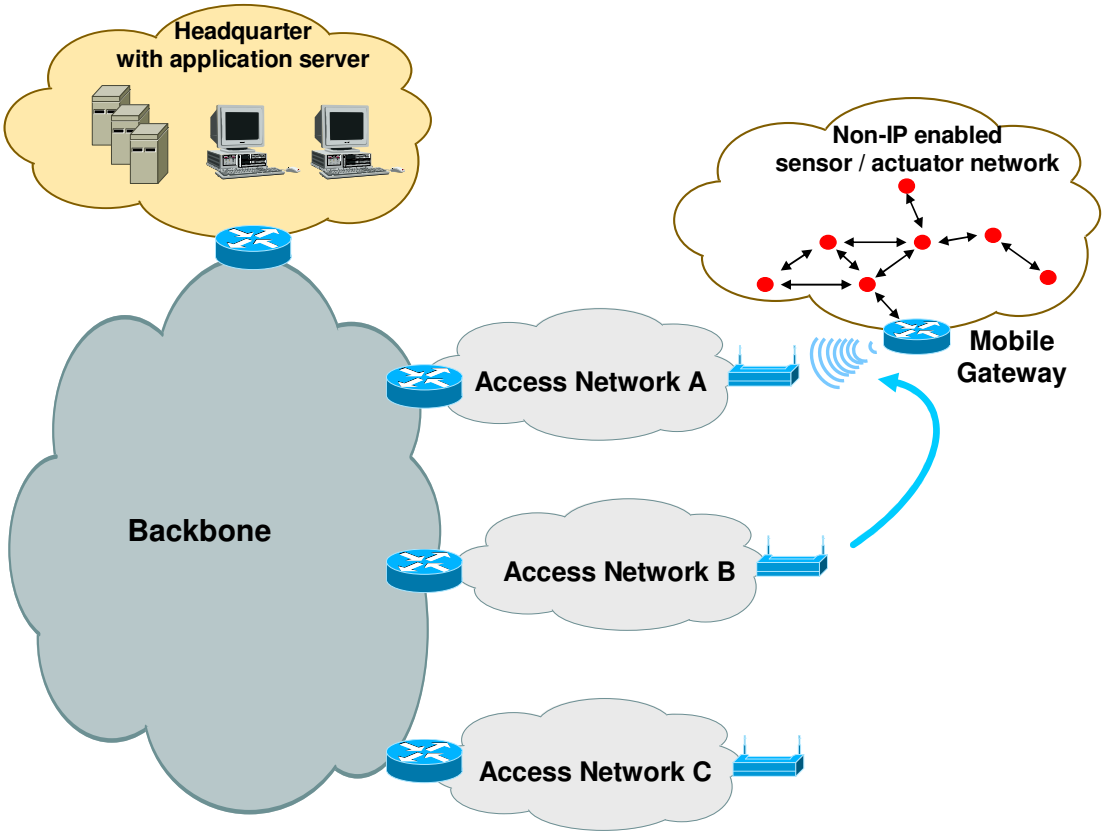


Figure 23: Mobile gateway connecting to non-IP enabled sensor / actuator network

A real-life example of such a scenario could be the fire in the road tunnel scenario investigated within the RUNES small-scale demonstrator. Here the sensor network could be sensors deployed in the road tunnel for smoke and temperature measurement, or sensors of a fire-fighters BAN. The actuators could be video devices installed in the tunnel or mounted on the fire-fighter's helmets, which can be zoomed and panned. The application servers partly could be local on a fire-truck, but could also be partly remote on a control centre connected to the Internet. The sensor / actuator networks may be connected to a fire-truck at the tunnel entry, which provides connectivity via 3G, satellite or WiMax hot spots. Additionally to the connection to the backbone the mobile gateway on the fire-truck also needs to collect sensor data and forward it encapsulated in IP packets further on to the control centre, and decapsulate control commands received from the controls centre and forward them to the respective actuators. This way the mobile gateway acts as a kind of application proxy for the sensor / actuator

network, and will be contacted by the remote control centre for each information exchange with the sensor / actuator network.

Besides this fire in the road tunnel there are plenty of other scenarios requiring a mobile gateway connecting a sensor / actuator network, such as mobile access router connecting a car's on-board network, a mobile phone connecting healthcare sensors of a BAN, or a train access router connecting the train network.

In all these scenarios one needs to maintain connectivity from remote applications to the mobile gateway, consequently one needs to deal with the mobile gateway changing dynamically its access points to the backbone. If this change of access points happens without changing the subnet technology itself, that is, if the mobile gateway e.g. moves between different 3G base stations or different WLAN access points, the gateway mobility often is handled already on the subnet level, without requiring changing the IP address assigned to the mobile gateway. Consequently the gateway mobility is transparent to the applications and the user.

However, at least when moving between different types of subnets the IP address of the gateway will change as well. In this case one needs to take care that the remote applications still can reach the mobile gateway at its current point of attachment. For this purpose one could use a fixed DNS name for the mobile gateway, and update dynamically the DNS entry with the respective IP address, however, this approach is not very scalable. Alternatively one can use Mobile IP technology for taking care of this issue. In the latter case the mobile gateway will get a permanent IP address allocated, which is taken from its usual home network, e.g. the network of a fire-department. At the respective access points a mobile gateway will then additionally get assigned a temporary remote IP address taken from the address network's address range. As described in [75] Mobile IP functionality could take care about shielding the dynamically changing remote IP address from applications and users, which will continue to communicate with the mobile gateway using its permanent home address.

For this purpose at least Mobile IP mobile node functionality needs to be integrated on every mobile gateway. Additionally a Home Agent needs to be deployed on each home network, e.g. on the fire-departments network. In order to benefit from advanced features like e.g. route optimization, it will be required to integrate Mobile IP correspondent node functionality in application servers.

Mobile IP functionality basically is available for IPv4 and IPv6, however, for the following reasons it is recommended to make use of Mobile IPv6 functionality as described in [73]:

- For the deployment of high numbers of mobile gateways, which could be the case with 3G handsets or car access router, a large address range will be required. This could be easier handled with IPv6.
- Contrary to Mobile IPv4 Mobile IPv6 more efficiently handles mobility, e.g. it integrates route optimization in the basic specification, and provides with return routability security measures for the Mobile IPv6 signalling messages.
- The support of network mobility as described in chapter 10.2 currently is only specified for IPv6 [74]. This network mobility support is mainly an extension of Mobile IPv6 functionality, consequently host and network mobility support functionality could be nicely integrated for IPv6.
- There is already ongoing work on integrating Mobile IPv4 and Mobile IPv6 functionality in a single signalling protocol called Dual Stack Mobile IPv6 (DSMIPv6) [76] to allow for easier IPv6 transition. DSMIPv6 as the name already indicates is based mainly on Mobile IPv6 functionality.

The kind of gateway mobility support outlined in this section is not only required for non IP enabled sensor / actuator networks, it could also be suitable for IP enabled sensor / actuator networks, which

shouldn't be allowed to receive IP connectivity from the backbone. For example due to security concerns one might want to restrict remote access to the gateway only, as this would be able to perform tasks like user and device authentication, service authorization or access control. Still the sensors and actuator could be IP enabled, and also communicate using IP with the gateway. However, there won't be an end-to-end IP connection between remote applications and the sensors / actuators, the mobile gateway in the middle will act as proxy. Consequently again only the gateway mobility needs to be dealt with in a way transparent for applications and users, and for this Mobile IP would be appropriate.

10.2 Mobility of gateways connecting IP enabled sensor / actuator networks

Figure 24 illustrates a scenario, in which an IP enabled sensor / actuator network will be connected via a mobile gateway to different access networks to an IP-based backbone network. Also connected to the IP backbone is a location hosting applications server, which e.g. require access to the data collected in the sensor network, or which issue control commands to the actuators.

However, this time sensors and actuators are IP enabled themselves, and also allow direct IP connection from the backbone, that is, between remote applications and sensors / actuators end-to-end IP connections will be established.

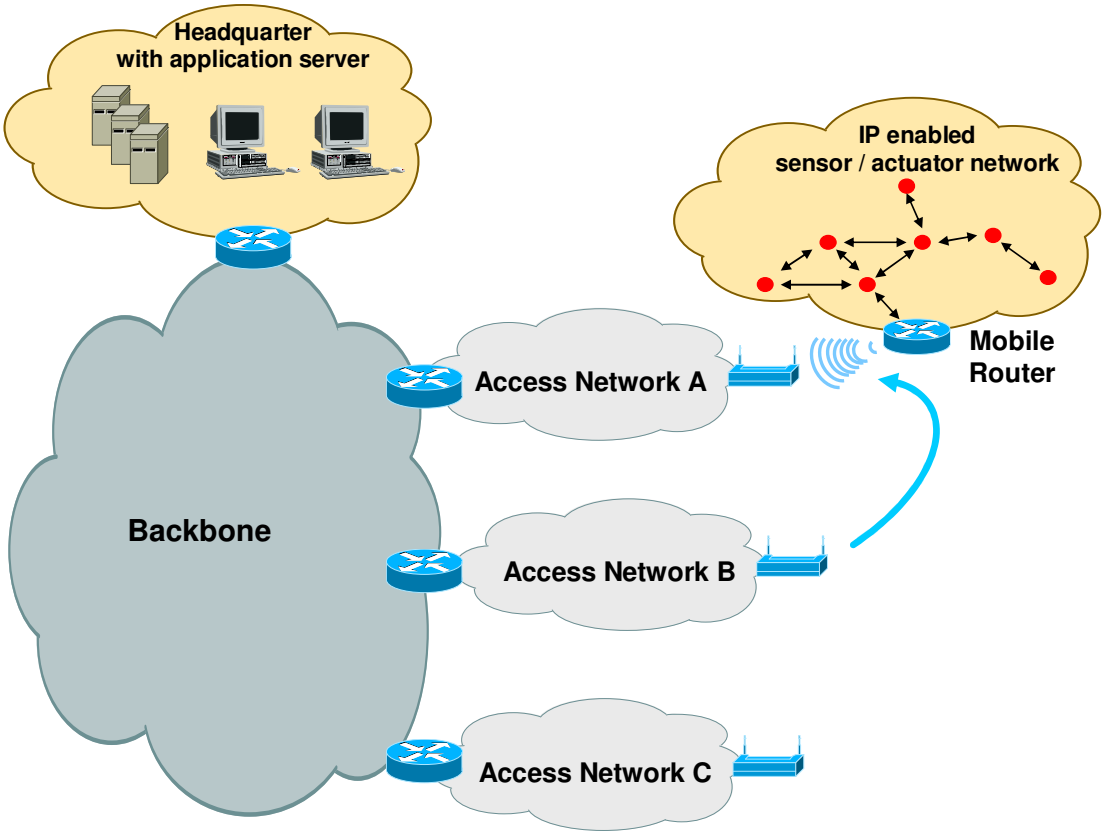


Figure 24: Mobile router connecting to IP enabled sensor / actuator network

Examples from real life for such a scenario are quite similar to the ones discussed in chapter 10.1, only the networking architecture will slightly differ.

Again connectivity from remote applications needs to be maintained during mobility, however, this time it is not only the connectivity to the mobile gateway itself, but the connectivity to each single sensor or actuator node. If this change of access points happens without changing the subnet technology itself, the mobility often will be handled already on the subnet level, without requiring changing the IP addresses assigned to the mobile gateway and the sensor / actuator nodes. Consequently the mobility is transparent to the applications and the user.

However, at least when moving between different types of subnets the IP address of the mobile gateway will change. In this case one needs to take care that the remote applications still can reach the mobile gateway at its current point of attachment, and via the mobile gateway also the connected sensor / actuator network. Using dynamic DNS would not help for this mobility scenario, as in this case the applications will not communicate with the mobile gateway, but with the sensors / actuators directly. Consequently it will ask DNS for the IP addresses of the respective sensors and actuators, and not for the dynamically changing IP address of the mobile gateway. Therefore the only way here is to make use of NEMO functionality for taking care of this issue in a way, transparent for user and applications.

Using NEMO as described in [74] the mobile gateway will act as the mobile router, the connected sensor / actuator network will represent the mobile network. The mobile gateway again will get on its interface external to the sensor / actuator network a permanent IP address allocated, which is taken from its usual home network, e.g. the network of a fire-department. At the respective access points a mobile gateway will then additionally get assigned a temporary remote IP address taken from the address network's address range. NEMO functionality will then take care about shielding the dynamically changing remote IP address of the mobile gateway from applications and users, which will continue to communicate directly with the sensor / actuator nodes using their permanent address in the mobile network.

For this purpose at least NEMO mobile router functionality needs to be integrated on every mobile gateway. Additionally a NEMO capable Home Agent needs to be deployed on each home network, e.g. on the fire-departments network.

It should be noted, that NEMO also supports the nesting of mobile networks, that is, that the mobile router of a mobile network attaches to the mobile network of another router. Such a kind of scenario could eventually become relevant in the fire in the road tunnel scenario. For example a mobile gateway on the fire-truck connecting the tunnel network to the backbone network could be regarded as the first level mobile router. To its mobile network, the tunnel network, connects now a second level mobile router, the mobile router of a fireman connecting his BAN. For access to / from the backbone network this second level mobile router of the fireman could roam between different fire-trucks located throughout the tunnel, which means, it will roam between their different mobile networks.

NEMO currently is only specified for IPv6, however, technically many of the reasons given in chapter 10.1 for an IPv6 based gateway mobility support solution applies here for network mobility support as well.

10.3 Conclusions and Future work

It is obvious, that more and more application scenarios will be able to benefit from sensor networks accessible from a Wide Area Network such as the Internet, consequently a gateway between the sensor network and the WAN is required. Part of these scenarios will benefit from sensor networks being able to move to different geographical locations; therefore their gateways need to adequately support this kind of mobility.

As the mobility support should be kind of transparent to the applications and not require significant resources from the sensor network itself, a solution is to integrate the mobility support functionality within the gateway. Which kind of mobility support will be most suitable, depends on the way the sensor network is exposed to the WAN. If it is visible only via the gateway acting as proxy, then a host mobility solution for the gateway seems adequate, if the sensor nodes themselves are directly accessible from the WAN, and network mobility solution would be better.

In future WP4 work, a design of a mobility support solution will be devised, which aims to provide the highest flexibility on mobility support mechanisms in order to serve as many scenarios as possible. For this design the selected method(s) of connecting the sensor networks to a WAN will be closely followed and considered.

11 Delay Tolerant Networking

There is a growing interest in the Internet community in Delay Tolerant Networking communication mechanisms as a means of implementing message delivery in extreme and performance-challenged network environments, where end-to-end connectivity cannot be assumed. In such environments, one should be still able to guaranteed delivery of messages to their destination in order to support operation of his/hers applications. Examples of such environments include spacecraft, military/tactical, some forms of disaster response, underwater, and some forms of ad-hoc sensor/actuator networks.

A number of techniques have already been proposed in the literature. These techniques aim at achieving delivery of messages to their destination in disconnected environments. We will categorise these techniques as following:

- Single-copy techniques (eg. dynamic Dijkstra algorithm) that need information of the dynamics of the network, eg. current available bandwidth of a node, queue size, etc.
- Multiple-copy techniques (i.e. techniques using replication of messages). These methods try to maximize the delivery probability to the destination node of the message by sending more than one replica of the message within the network, whereby each replica may follow a different route to the destination.

Epidemic routing [32] for mobile ad-hoc networks falls in the second category. As two nodes meet, they exchange the messages they keep in their buffers, and; hopefully, the destination node will meet with a node holding the messages destined to this particular destination. The problem with epidemic routing is that it does not take into account the buffer size of the nodes, i.e. many copies of the same message are kept into different nodes at the same time and as such, epidemic routing can not directly apply in scenarios where the resources are scarce, as is the case of sensor networks studied within the RUNES project.

An alternative to the Epidemic routing protocol is the Prophet protocol proposed in [34]. This protocol works by exchanging the metric “delivery predictability” between nodes, as they meet. The metric “delivery predictability” denotes the probability of encountering a certain node and is used at every node in order to decide whether or not to forward a message to a specific destination. The problem with this method is that it is difficult to calculate the delivery predictability metric in a systematic way.

In a resource-constrained environment, such as RUNES sensor network, where sensors have finite and small-size buffers, there is the need to reduce as much as possible the number of replicas in the network. Purpose of our work is to address this issue by designing and implementing a protocol that takes into account the “{m,n} hop” metric, which is used to specify how close a message is to its source (variable m) and at the same time, how close the message is to its destination (variable n). We have chosen this metric taking into account the following observations:

- if a message is close to the destination, it is likely that it will be delivered, so a copy of the message must be stored within the node;
- if a message is close to its source, a copy of the message must be kept, otherwise, if the copy of the message is deleted, the message may never reach its destination as the number of nodes holding the same copy of the message is small (i.e. all the nodes close to the source that hold a copy of the message).

Therefore, the “hop” metric we will use in our work is the vector {m,n}, where the variable m denotes the number of hops from the source and the variable n the number of hops from the destination.

The rest of the paper is organised as follows. Section 2 presents and categorises related work to the work presented here, section 3 provides details of our new algorithm and section 4 experimental

results from simulation of the algorithm. Section 5 concludes the paper and outlines direction for future work and research within the RUNES project.

11.1 Related Work

A comprehensive overview of related work to our efforts can be found in [24]. Below, we list and categorise the most important techniques, as found in [24], addressing the problem of message delivery in disconnected environments.

11.1.1 Store and Forward Systems

Electronic mail may appear as a natural approach for handling message delivery in a frequently disconnected environment. It does provide the required store-and-forward capability, but typically lacks any robust approach to dynamic and multi-path routing. In the Internet, successful e-mail exchange (i.e. with SMTP) generally depends both on a successful DNS request/response transaction along with a reliable transport layer exchange protocol. The mail exchanger DNS facility ("MX record") provides a limited form of routing in the e-mail overlay, but falls far short of handling the type of the problem discussed in this paper. Thus, e-mail (at least Internet e-mail) is unable to deal with true dynamic routing and is not very robust to errors during message transfer.

Prior to the wide availability of Internet connectivity, the UUCP network carried news and electronic mail to a small but growing (and active) user community. It used source routing, and responsibility for route selection was left primarily to end-users. It was later enhanced with a form of static routing based on computing shortest paths on a topology graph distributed by e-mail (or entered manually. A user program called *pathalias* performed this function). To choose among local outgoing connections, fixed costs were assigned manually to links based upon the frequency and the quality of the connection. The Minimum Expected Delay (MED) algorithm presented in [24] is similar to above idea.

11.1.2 Optimization Techniques and Network Flows

The field of operations research is rich with variations on optimization problems involving network flows, shipment of materials, and scheduling. A complete survey could not possibly be accomplished here. However, some of the more relevant work includes the quickest transhipment problem, dynamic multi-commodity flow problems, etc. [25]. Frequently, network flow problems that involve temporal sensitivities are solved as traditional graph problems on time-expanded graphs. These graphs, introduced as early as 1962 by Ford and Fulkerson [26], can be used to capture the temporal dynamics as additional nodes and edges. The issue with computing on such graphs is that they can significantly expand the search space, leading to very large problems.

Focusing specifically on shortest path (single path) solutions, algorithms for dynamic networks have been investigated by Orda et.al. in [27]. In this respect, our primary contribution has been to define appropriate cost functions in the context of the DTN routing problem. The issue of splitting (allowing message fragments to take multiple paths) and buffer constraints makes the problem much more challenging. The problem is further complicated because of multiple commodities, time-varying capacities, non-zero and possibly time-varying propagation delays, etc. The specific case of zero propagation delays with finite buffers and time-varying capacities has been addressed by Ogier in [28].

11.1.3 Routing in Disconnected Mobile AdHoc Networks

Most of the work exploring the DTN routing problem [24] is focused on cases where the topology dynamics are known (or nearly known) in advance. Clearly, however, many systems will not exhibit such predictability or will exhibit partial predictability. A series of efforts [29][30][31][32] in the context of sensor/mobile-ad/hoc networks looks into providing connectivity when topology dynamics are unknown. Generally, these techniques employ a form of data duplication within the network and achieve eventual delivery. Such duplication requires a way of getting rid of unnecessary copies to reduce the buffer occupancy. We believe such an approach, possibly employing some of the epidemic techniques in conjunction with the techniques discussed here, may be appropriate for networked embedded systems envisioned within the RUNES project.

11.1.4 Delay Tolerant Networking Routing Techniques

The routing objective of traditional routing schemes has been to select a path, which minimizes some simple metric (e.g. the number of hops between the source and the destination of the message). For DTN networks, however, the most desirable objective is not immediately obvious. One natural objective is to maximize the probability of message delivery. Messages could potentially be lost due to creation of a routing loop or the forced discarding of data when buffers are exhausted in resource-constrained environments, such as sensor networks.

Furthermore, while DTN applications are expected to be tolerant of delay, this does not mean that they would not benefit from decreased delay. Simulation results presented by Jain, Fall et al. in [24] show that minimizing delay lowers the time messages spend in the network, reducing contention for resources (in a qualitative sense). Therefore, as [24] observes, lowering delay indirectly improves the probability of message delivery.

11.1.5 Proactive Routing vs. Reactive Routing

In proactive routing, routes are computed automatically and independently of traffic arrivals. Most Internet standard routing protocols and some ad-hoc protocols such as DSDV (Destination Sequenced Distance Vector) and OLSR (Optimized Link-State Routing) are examples of this style [33]. In a DTN, these protocols are capable of computing routes for a connected sub-graph of the overall DTN topology graph. They fail when they are asked to provide paths to nodes, which are not currently reachable. Despite this drawback, proactive network-layer routing protocols may provide useful input to DTN routing algorithm by providing the set of currently-reachable nodes from which DTN routing may select preferred next hops.

In reactive routing, routes are discovered on-demand when traffic must be delivered to an unknown destination. Ad-hoc routing protocols such as AODV (Ad-hoc On-demand Distance Vector) and DSR (Dynamic Source Routing) are examples of this style [33]. In these systems, a route discovery protocol is employed to determine routes to destinations on-demand, incurring additional delay. These protocols work best when communication patterns are relatively sparse. For a DTN, as with the proactive protocols, these protocols work only for finding routes in a connected sub-graph of the overall DTN routing graph. However, they fail in a different way than the proactive protocols. In particular, they will simply fail to return a successful route (from a lack of response), whereas the proactive protocols can potentially fail more quickly (by determining that the requested destination is not presently reachable).

In a DTN, routes may vary with time in predictable ways and can be pre-computed using knowledge about future topology dynamics. Employing a proactive approach would likely involve computing several sets of routes and indexing them by time. The associated resource requirements would be prohibitive unless the traffic demand is large and a large percentage of the possible network nodes exchange traffic. Otherwise, a reactive approach would be more attractive.

A related issue is route stability, a measure of how long the currently known routes are valid. Route stability depends on the rate of topological change. With relatively stable routes one can employ route caching to avoid unnecessary routing protocol exchanges. With future knowledge about topology changes, caching could be especially effective in a DTN because it may be possible to know ahead of time exactly when to evict existing cached route entries.

11.1.6 Source Routing versus Per-Hop Routing

In source routing the complete path of a message is determined at the source node, and encoded in some way in the message. The route is therefore determined once and does not change as the message traverses the network. In contrast, in per-hop routing the next-hop of a message is determined at each hop along its forwarding path. Per-hop routing allows a message to utilize local information about available contacts and queues at each hop, which is typically unavailable at the source. Thus, per-hop routing may lead to better performance. Unfortunately, due to its local nature, it may lead to loops when nodes have different topological views (e.g. due to incomplete or delayed routing information).

11.1.7 Message Splitting

A message is split when forwarded in such a way that different parts (fragments) are routed along different paths (or across different contacts on the same path). This technique may reduce the delay or improve load balancing among multiple links. It is particularly relevant in DTNs because messages can be arbitrarily large and may not fit in a single contact. However, splitting complicates routing because, in addition to determining the sizes of the fragments, we also have to determine corresponding paths for the fragments.

11.1.8 A Routing Evaluation Framework

In their work [24], the authors formulate the DTN routing problem when the connectivity patterns are known, then provide a framework for evaluating various routing algorithms, and finally show a simulation-based comparison of several of their own algorithms. They also include an optimal algorithm based on a linear programming approach to serve as a basis for comparison with the simulations.

As formulated in [24], the DTN routing problem has many input variables such as the dynamic topology characteristics and the traffic demand. Complete knowledge of these variables facilitates the computation of optimal routes. However, with partial knowledge, the ability to compute optimal routes is hampered, and the performance of the resultant routing is expected to be inferior. To understand this fundamental trade-off between performance and knowledge, the authors define a set of abstract knowledge oracles, each able to answer questions one could ask. These oracles are notational elements used to encapsulate particular knowledge about the network required by different algorithms. A key objective of their evaluation framework is to understand the relationship between algorithm performance and the use of these oracles.

Figure 25 illustrates this conceptually by showing the expected performance and oracle requirements for each proposed in [24] routing algorithm. The x-axis depicts the amount of knowledge (increasing in the positive direction). The y-axis depicts the expected performance that can be achieved using a certain amount of knowledge. The figure shows that more knowledge is required to attain better performance. Labels on top show algorithms developed in [24] using the corresponding oracles.

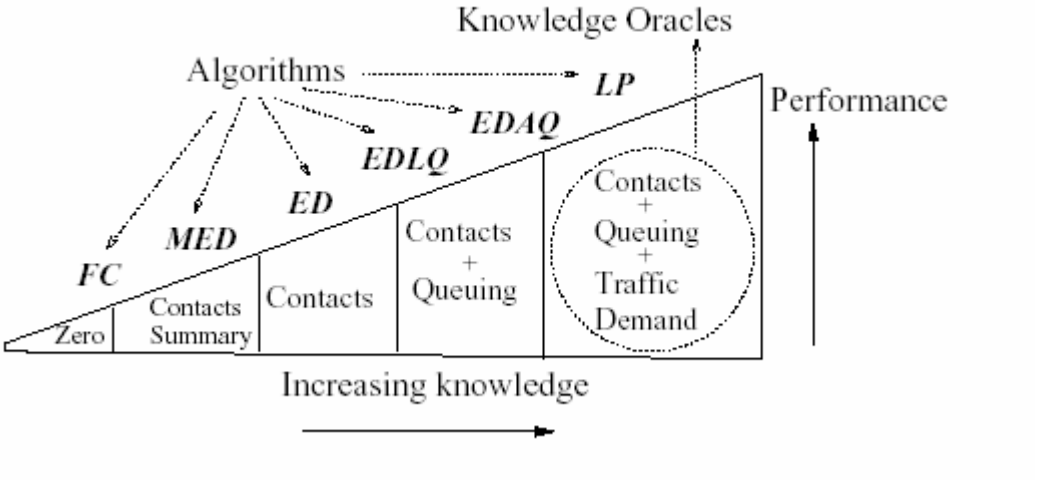


Figure 25: Conceptual performance vs. knowledge trade-off (from [24])

11.2 RUNES Approach: Constrained Replication Using Source and Destination Based Routing

Each DTN gateway will be holding a table of {m,n} distances for DTN gateways it knows about. As two gateways meet, they will be exchanging their distance tables and will update their tables accordingly. Consider Figure 26 as an example for the topology of DTN gateways.

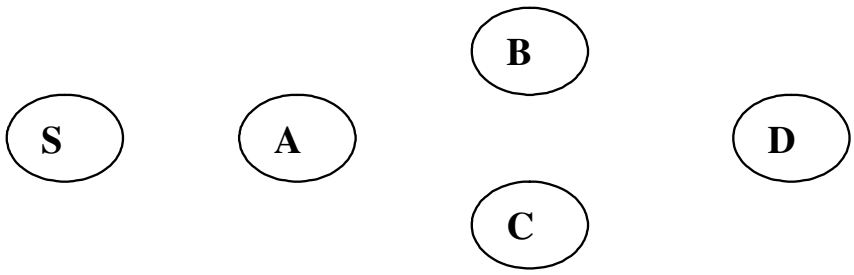


Figure 26: Example of a DTN topology

Assume that A meets B and C. This means that A longer knows that the distance to B and C is 1. The same is valid for B and C: B knows that it is one hop away from A and C now knows that it is one hop away from A. We can represent this knowledge in a two dimensional table, as following:

A table:
D[A][B] = 1
D[A][C] = 1
D[C][A] = 1
D[B][A] = 1

B table:
D[B][A] = 1
D[A][B] = 1

C table:
D[C][A] = 1
D[A][C] = 1

Assume now that A meets S. This means that A and S will exchange their tables and each one will update its local tables. Thus, at the end of this exchange phase A will have updated its table as follows:

D[A][S] = 1
D[S][A] = 1
D[S][B] = D[S][A] + D[A][B] = 2
D[S][C] = D[S][A] + D[A][C] = 2

From the previous phase:

D[A][B] = 1
D[A][C] = 1

S will update its table accordingly.

When A has updated its table, it should propagate the new table values to its neighbours if possible, i.e. to B and C. If this is not possible, the propagation will happen the next time it meets with each of its neighbours. This means that if A and B are still in contact, B will update its table as following:

D[S][B]= D[S][A] + D[A][B] = 2
D[A][B] = 1
D[S][C] = 2 (this information comes directly from A)
D[A][C] = 1 (…)

Assume now that B meets D. The table information exchange will result in the following table within B:

D[B][D] = 1
D[S][D] = D[S][B] + D[B][D] = 2 + 1 = 3

The table of D will be:

D[D][B] = 1
D[D][S] = D[D][B] + D[B][S] = 1 + 2 = 3

11.3 Decisions based on the {m,n} values

The {m,n} values can be used to decide to check whether a node is close to either the source or the destination of the message. This in turn affects the choice of how the message is forwarded (or dropped).

Assume that a message is created by S. The destination of the message is D. The message is copied to A when S is in contact with A. From the table exchanges phases, A knows that D is 2 hops away if the message is sent to B. C does not know anything about D, so the table value D[C][D] = infinite. Therefore, a good choice is that A will prefer B instead of C, and thus A will copy the message only to B.

Another choice would be to copy the message to both B and C, taking into account that both B and C are only two hops away from S. This implies that it is possible that in the future, C will find a path to send the message to D.

In a general case, assume that a S wants to deliver a message to D. Assume now that the message has travelled until node K and K wants to decide on what it should do with the message that has come to it from S. Figure 27 depicts this scenario.

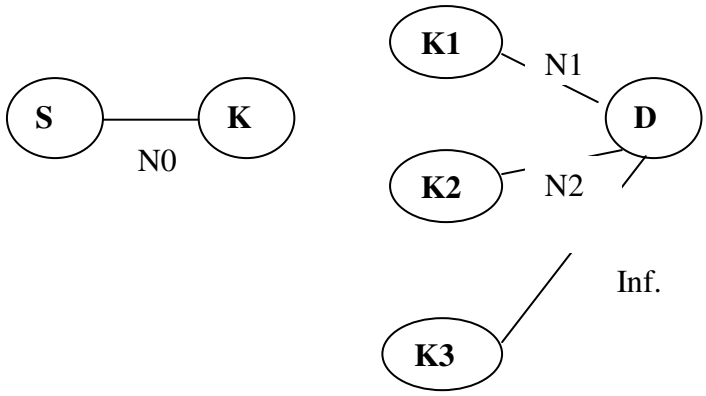


Figure 27: General DTN scenario

Assume that the K knows that:

D[S][K] = N0
D[K1][D] = N1 where K1 is a neighbour of K
D[K2][D] = N2, where K2 is also a neighbour of K
D[K3][D] = inf, where K3 is also a neighbour of K

Several choices can be made by the gateway K taking into account the number of hops of the message from its source and its destination (i.e. the {m,n} values).

11. If N0 is large (above a maximum value) and both N1 and N2 are large, this means that the message is somewhere between the source and the destination, neither close to the source, nor close to the destination. In this case, if the buffer space of K is full, K may want to drop the message from its message queue (or drop some older messages experiencing the same problem).
12. If N0 is small and both N1 and N2 are large, this means that the message is close to the source, so K should not delete the message. A choice would be to copy the message only to K1 and K2, since K3 has never met D before. Another more resource efficient choice is to

send the message to K1 if N1 < N2, as K1 is closer to the destination, and such; K1 is more likely to deliver the message to D.

13. Another option could be to send the message to K1 with probability N2/N1+N2 and to K2 with probability N1/N1+N2. This means that some messages will be travelling from K1 and other messages from K2, but more messages will be routed from K1 if N1 < N2 (i.e. because K1 is closer to the destination than K2, so K1 is more likely to deliver the message to destination D).

11.4 Experiments and Results

In order to test our algorithm, we have extended the Java-based DTN simulator [104] with additional modules, implementing the constrained replication algorithm that we described in the previous sections of this chapter.

In its current version, the DTN simulator [104] supports a limited set of routing algorithms, which are the following: Random Routing, Minimum Hop Count Routing and Earliest Delivery Routing. In our implementation, the DTN simulator contains additional components, which implement the Epidemic Routing and our Constrained Replication algorithms. We also extended the simulator with additional modules that measure the queue sizes of the DTN gateways and produce buffer occupancy statistics.

In the following, we will show results from an example usage of our simulator. Figure 25 shows the network topology we used to conduct a set of two experiments. In the first experiment, the Epidemic Routing algorithm is used, where each DTN gateway sends a copy of the message to every member of its neighbours set, as they meet during the simulation. The configuration file used for specifying the topology in Figure 25 and the other simulation parameters (eg. Buffer sizes, time of contacts, bandwidth of links, traffic generators, etc.) is the file shown in Figure 26.

Using the epidemic method, as the results in Figure 27 show, messages achieve a good delivery probability, but all nodes in the system that participate in the delivery of the message from **Node 0** to **Node 3**, have a high buffer occupancy (2000 units is the buffer size of every DTN node), as all neighbours of a node holding the message are flooded with the same copy of the message, as they meet the holder of the message.

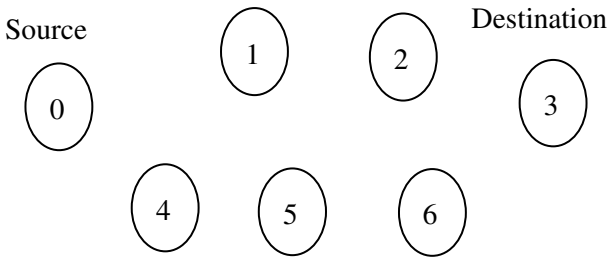


Figure 28: Simulation Topology

```
# The following commands are read by the DTN simulator. The first entry in each
#command is the time at which that command will be executed by the simulator

# some identification for this simulation run
0 set -namekey  leo.scen

# simulation time  in tics
0 set -simtime  86400

# some random seed
0 set -random   22

#
#      "0" ------ 1 ------- 2 ------- "3"
#          ----   4 ----  5 --6------
#
#

# TOPOLOGY

#create first node with id "0"  and buffer size 2000 units
0 create_node 0 2000
0 create_node 1 2000
0 create_node 2 2000
0 create_node 3 2000
0 create_node 4 2000
0 create_node 5 2000
0 create_node 6 2000


# CONTACTS


# at time 100.0 create a contact, id value is given -1. this will allow simulator to
#generate its own id for keeping track of contact.

#now the format for creating contact is:
# [time, "create_contact", contact_id, source_node_id, dest_node_id, bandwidth,
#delay, start_state, up_distribution, down_distribution, ?bidir]

#start_state can be: up or down,
#up_distribution, down_distribution: is a way of defining a sequence of #numbers.
#see util/Probability_Distribution.java for its format
# in the end the "bidir" can be used to indicate that it is a #bidirectional link. a
#bi-directional link is equivalent of saying that #there are two links with the same
#pattern. if nothing is specified link #is assumed to be unidirectional

#D_20 stands for the fact that the link is up for length of 20 every time its up. D
#stands for deterministic.
#if E_20 is used instead then the sequence of length of times for which the link is
#up would be derived form an exponential distribution with mean 20.
# explicit sequence of numbers can also be specified through a file. see
#util/Probability_Distribution_File.java

100.0 create_contact    -1  0 1   500  0.07 down   D_100 D_100
100.0 create_contact    -1  1 2   500  0.09 down   D_250 D_250
100.0 create_contact    -1  2 3   500  0.09 down   D_450 D_450

100.0 create_contact    -1  0 4   500  0.09 down   D_100 D_100
100.0 create_contact    -1  4 5   500  0.09 down   D_250 D_250
100.0 create_contact    -1  5 6   500  0.09 down   D_450 D_450
100.0 create_contact    -1  6 3   500  0.09 down   D_450 D_450

# TRAFFIC from 0 ---- > 3
#Creating a CBR agent

101.0 create_agent  cbr 0.01 0 3 1000
```

Figure 29: Simulation configuration file for all routing schemes

"cmd: ./dtn scen/leo.scen flood" (leo.scen is the file shown in Figure 26)

```
Called ROUTING_ALGO at node 0. Next node(s) for the message863 is
 [ contact id 2:Nd:0 -Nd:1 ,  contact id 6:Nd:0 -Nd:4 ]

FLOOD: Forwarding using contact  contact id 2:Nd:0 -Nd:1
FLOOD: Forwarding using contact  contact id 6:Nd:0 -Nd:4
CURRENT Queue size of NODE ided :(0) =0.0

 FR attributes leo.scen_cost_flood  FR generated  864000.0   received  = 782000.
0   recv/generated = 0.905   dropped = 80000.0   drop probability = 0.092

STATS: Printing now AVERAGE queue sizes for ALL NODES
Samples of NODE ided :(6) =390
AVERAGE Queue size of NODE ided :(6) =317.94871794871796
---------------------------------------------
Samples of NODE ided :(5) =390
AVERAGE Queue size of NODE ided :(5) =689.7435897435897
---------------------------------------------
Samples of NODE ided :(4) =430
AVERAGE Queue size of NODE ided :(4) =802.3255813953489
---------------------------------------------
Samples of NODE ided :(3) =0
AVERAGE Queue size of NODE ided :(3) =0.0
---------------------------------------------
Samples of NODE ided :(2) =390
AVERAGE Queue size of NODE ided :(2) =715.3846153846154
---------------------------------------------
Samples of NODE ided :(1) =430
AVERAGE Queue size of NODE ided :(1) =802.3255813953489
---------------------------------------------
Samples of NODE ided :(0) =863
AVERAGE Queue size of NODE ided :(0) =500.57937427578213
---------------------------------------------
FR Exitting main at simulation time 86401.0
Sat Jan  7 21:15:48 GMTST 2006
```

Figure 30: Simulation log file when using the epidemic algorithm

In the second experiment, we used again the configuration file shown in Figure 26, but this time the algorithm used to route messages is our constrained replication algorithm. The policy that we used in this specific experiment is that messages are sent to the neighbour with the minimum **n** value (i.e. the minimum number of hops to the destination). The results of the simulation are shown in Figure 28.

"cmd: ./dtn scen/leo.scen rep" (leo.scen is the file shown in Figure 26)

```
Routing ALGO + 1
Called ROUTING_ALGO at node 0. Next node(s) for the message863 is
[ contact id 1:Nd:0 -Nd:1 ]
CURRENT Queue size of NODE ided :(0) =1000.0

 FR attributes leo.scen_cost_rep  FR generated  864000.0   received  = 534000.0
  recv/generated = 0.618   dropped = 328000.0   drop probability = 0.379

STATS: Printing now AVERAGE queue sizes for ALL NODES
Samples of NODE ided :(6) =0
AVERAGE Queue size of NODE ided :(6) =0.0
---------------------------------------------
Samples of NODE ided :(5) =0
AVERAGE Queue size of NODE ided :(5) =0.0
---------------------------------------------
Samples of NODE ided :(4) =0
AVERAGE Queue size of NODE ided :(4) =0.0
---------------------------------------------
Samples of NODE ided :(3) =0
AVERAGE Queue size of NODE ided :(3) =0.0
---------------------------------------------
Samples of NODE ided :(2) =533
AVERAGE Queue size of NODE ided :(2) =611.6322701688555
---------------------------------------------
Samples of NODE ided :(1) =689
AVERAGE Queue size of NODE ided :(1) =753.2656023222061
---------------------------------------------
Samples of NODE ided :(0) =863
AVERAGE Queue size of NODE ided :(0) =1001.1587485515643

FR Exitting main at simulation time 86401.0
Sat Jan  7 21:25:06 GMTST 2006
```

Figure 31: Simulation log file when using the constrained replication algorithm

As the results in Figure 28 show, the constrained replication algorithm with the policy "choose the neighbour with the minimum **n** value", achieves worse delivery probability (61.8%) than the epidemic method (90.5%), as only one path of routers ("0-1-2-3") is used to deliver the message to the destination (Node 3 in Figure 25). However, our method results in a less much less buffer occupancy within the DTN network, as only one path of routers is used to deliver the message to the destination node.

Note that this experiment is only an example experiment, where a quite simple policy controls the behaviour of the algorithm. Additional and more complex policies can also be implemented in order to tweak the behaviour of the algorithm in order to achieve a higher delivery probability in a number of scenarios different that the one chosen here (Figure 25). The reason is that the simple scenario chosen here uses a small number of nodes, with only two paths delivering the message to the end node. We plan to work on more complex scenarios in the remaining of the project and to evaluate different policies that impose the constrained replication behaviour of our algorithm.

11.5 DTN Gateways

Earlier in this discussion we for clarity made the simplification that DTN gateways always consist of one node only. In reality, this will not always be the case. In our target scenarios we envision systems consisting of many and heterogeneous nodes. The nodes that the system is built from will have different radios, widely differing storage capacities, different energy budgets etc. A possible view is that the DTN gateway consists of a federation of more or less permanently associated devices. In effect the DTN gateway is built from the same pieces and resources that the system is built from. A possible example is a wireless sensor network used to monitor animal motion in a forest. To the north and south of the forest there are roads with DTN capable trucks that infrequently pass. Data must be replicated both on the south and north side in order for it to be transferred during the brief period when trucks pass by. A different example includes a person with a personal area network (PAN). The interior of the PAN is connected using Bluetooth and consists of (amongst other gadgets) a phone with GSM connection and a MP3-player with a WiFi-interface. If a WiFi hot-spot is encountered, it will be used to transfer the bundles, otherwise GSM is used.

An important observation is that the interior of a federation may not always be permanently connected, inside the federation many nodes may be sleeping a large fraction of the time and this must be taken into account when managing communication and storage resources. In order to manage energy resources different nodes may be temporarily designated the task of tracking and making use of contact opportunities.

As discussed earlier the role of a DTN gateway is to keep track of contacts (opportunistic, predicted or scheduled) and to make use of these when forwarding. Other responsibilities of the gateway is to manage the various resources that the gateway consists from. For instance it may be necessary to relocate bundles so that they are near a node with a particular radio technology or near a node that is positioned at a particular point. Bundles may also have to be relocated in order to prevent queues from overflowing or to "route around" spots that are low on energy.

11.6 Infrastructure

It is reasonable to expect that at least some DTN applications will exists in an environment where connectivity to the Internet or other local networks happens periodically. This is certainly the case with people and vehicles moving around in a urban environment encountering WiVi hot-spots and may also be true in other scenarios. The existing infrastructure can be augmented with additional

functionality that boosts the effectiveness of DTN routing protocols by routing via large and well connected networks with plenty of resources.

One way of realising this would be to create an overlay over the Internet that acts as a publish/subscribe substrate for bundles. As roaming gateways come in contact with the Internet they "publish" bundles they would like to route via the Internet. At a later point a different DTN gateway will retrieve the bundles using the "subscribe" feature. Because the network is rich in resources we can afford to move bundles around to where they will be picked up, consequently it will often be the case that bundles are left in one place of the network and retrieved somewhere completely different.

11.7 The Bandwidth Mismatch Problem

Consider a car driving past a WiFi hot-spot. The hot-spot can provide bandwidth on the order of 50Mbit/s but the car will only be able to use this bandwidth for a few seconds. If the hot-spot is connected to the Internet via ADSL the available bandwidth to the Internet will be limited to something like 500kbit/s, i.e. the car will only be able to utilise a small fraction of the locally available bandwidth. To deal with this problem it should be possible to provide hints to the overlay so that bundles can be transferred to anticipated points of contacts in advance. If we know 100 seconds in advance that the car will be arriving at a certain hot-spot it is possible to move 50Mbits of data there in advance, later when the car arrives the same amount of data can be retrieved in only one second.

It can easily be imagined how this approach can be part of a system solution for cars, bus routes, and trains. Vehicles don't move randomly and it should be possible to accurately predict possible points of contacts well in advance. The same approach can also be applied to human mobility. We routinely do the same little travels every day. We leave our kids at kindergarten, go to work, have lunch, go back to work, go shopping, and go home (or variations thereof). It is possible to detect these routine traversals of hot-spots and make judicious predictions about future (scheduled) contacts. The converse may also be true in some circumstances, if you are moving down the road, the hot-spots may predict your movements and inform the overlay so that bundles are transferred in anticipation.

11.8 Transfers inside a DTN capable sensor network

When we are moving bundles inside a sensor network there are two extremes for how this can be done, we may either move the entire bundle hop by hop, or we may move the bundle as a stream between the source and destination nodes. Sensor nodes often don't have much memory and buffering capacity, thus we will usually have to resort to the latter approach. Unfortunately this implies that we need to keep all the intervening nodes awake during the transfer and this consumes energy. To keep energy consumption to a minimum it is important to minimise the transfer time and to minimise the number of retransmissions. To deal with this problem we propose the use of DTC [118] distributed TCP caching.

Distributed TCP caching avoids end-to-end retransmissions by caching TCP segments in the network and performing local retransmissions. Ideally, each node would cache all the segments and the last node to have successfully received a segment would perform a local retransmission. Because sensor nodes are memory constrained, however, each node is only permitted to cache a small number of segments. Nodes therefore attempt to identify and cache segments that may not have been received successfully by the next hop, as indicated by a missing link layer acknowledgment.

In our simulations, we have assumed link layer acknowledgments. A TCP segment that has been forwarded but for which no link layer acknowledgment was received is assumed to have been lost in transit. Therefore, the segment is cached and will not be replaced with a segment with a higher sequence number. A cached segment is removed from the cache when a matching TCP acknowledgment is seen or when the segment times out.

DTC uses ordinary TCP mechanisms to detect packet loss: timeouts, duplicate, and selective acknowledgments and uses these to avoid useless end-to-end retransmissions. Every node participating in DTC needs to maintain a soft TCP state for each end-to-end connection for which it forwards packets.

We have performed simulations that show this mechanism to substantially improves TCP performance in wireless sensor networks: DTC significantly reduces the overall number of TCP segment transmissions and the number of end-to-end retransmissions. Analytical results also suggest that DTC performs well: The number of transmitted segments is only about 10% higher than would be necessary given an ideal caching strategy for short paths (six hops) and about 25% higher for longer ones (11 hops).

11.9 Conclusions and Future Work

In the previous section, we demonstrated the operation of our new algorithm in simulation. A lot of work will be undertaken in the next few months of the project, in order to evaluate our algorithm in complex networking scenarios. The results of this work will be presented in the next deliverable of the project. We also plan to implement the DTN protocol and our algorithm for the Tmote Sky motes, using the Contiki or the TinyOS operating system. Future work also will integrate our routing components with the middleware system developed by WP5 of the project.

Apart from the implementation work described above, we plan to work on a few research issues related to the work presented here. The following is a list of possible future research directions.

- Adaptive behaviour of the replication algorithm. Currently, the algorithm is configured by one policy in order to decide who and how replication operates within the DTN network. However, it would be possible that the policy controlling the behaviour of the algorithm is changed at runtime, according to monitoring information received from the network. This means that if we discover that a specific policy is not effective for a specific scenario, then a different policy can be chosen in order to deliver a better performance of the algorithm. This would require an adaptive management system for the policies, as the one presented in the PhD thesis [105].

- Implementation of an intelligent system to generate new configuration policies of the algorithm. Work in this direction has been addressed by AI research groups and we could investigate how to port their ideas in our context, i.e. try to see how we can generate, based on the system's history and current behaviour, new policies that tailor the behaviour of the algorithm in the current networking environment where the algorithm operates.

Regarding the DTN gateways research issue described earlier in sections 11.5 – 11.8, as a next step we should implement and evaluate DTN gateways that match our experimental platform, i.e. we need some kind of featherweight DTN implementation for our motes and a matching implementation that is attached to a traditional IP network. The latter part can for instance be implemented on a standard PC or laptop.

Furthermore, Distributed TCP caching (DTC) should be implemented in our motes; and, its effectiveness should be experimentally verified.

12 Conclusions and Future work

In this paper, we have investigated a number of networking solutions, which will help to create an efficient sensor networking architecture for the scope of the RUNES project. As a case study scenario, we introduced a sensor network architecture for road tunnels that is able to monitor various physical parameters of the tunnel and helps rescue personnel and passengers in case of emergency.

As far the road tunnels scenario is concerned (the road tunnel scenario has been selected as the demonstration scenario for the RUNES project), our sensor network architecture is most of the time wired; therefore the sensors residing in the infrastructure have endless power most of the time. In case of an emergency, where some wires are broken and some system nodes are demolished; the concerned part of the network switches to radio communication. Moreover; we have used a hierarchical approach in the infrastructure, and have proposed an overlay ad-hoc network over the sensor network. However, neighbouring nodes in the overlay ad hoc network have also direct communication between each other; thus saving the energy of the under laying sensor network.

In the rest of the RUNES project, we plan to integrate different solutions in our generic network architecture. This means that different addressing, routing, localization, security and mobility management schemes will co-exist in the RUNES network architecture, either in parallel or on top of each other.

Furthermore, from a RUNES perspective, we need to evaluate existing addressing and routing schemes against a number of key networking requirements. If our evaluation task results in the conclusion that existing solutions can not cater for the RUNES requirements, we will consider modifying existing addressing and routing schemes or designing and implementing new ones. We also need to define how to support functional and geographical addressing and routing within a RUNES sensor domain. Those dedicated addressing and routing schemes will be part of an overlay structure. Addressing is also effected by the security mechanisms that might be required from a privacy and integrity point of view. For example, special authentication mechanisms may be required.

All of the described autoconfiguration approaches are still under investigation; therefore at the time of this writing there has been no strong indication about which of them will be the best choice for RUNES. This requires further investigation, development, and testing work in order to enable autoconfiguration with RUNES sensor networks.

Regarding the research issue of localization, we analyzed in this paper the localization requirements in the RUNES architecture, reviewed the state-of-art of sensor network localization techniques, and proposed a number of localization mechanisms that are may be suitable for various RUNES application scenarios. The approach under investigation is to apply RF signal sensing to establish the location information of the ad hoc sensor nodes. Though RF sensing may not be accurate in an indoor or tunnel environment, pre-planning can often establish the signal patterns for location references. We also plan to evaluate the UWB, Ultra-sound and IR sensing and badge technology in order to see whether they can be applicable in the tunnel environment where the end of the tunnel would be much similar to an outdoor atmosphere with possibly direct exposure to sunlight. Filtering technology is to be applied in optimizing the output location results with multiple inputs from RF sensing and from Dead Reckoning (DR) system. Such a hybrid solution is expected to outperform both the sensing based solution and an isolated DR system.

Regarding ad hoc and sensor network integration, we identified lots of topics to be investigated in the future. First it should be investigated how to route and deliver unicast packets between the sensor and ad hoc network nodes. This implies using some kind of hierarchical routing mechanisms. It is also a question how to handle multicast and how the multicast groups are formed. It is possible to build up a multicast tree in advance, or it can be done on demand when necessary. Broadcasting has to be

analysed, as well. Since the ad hoc network above the sensor network can deliver the broadcast messages directly to nodes involved in the ad hoc network, the question is how far a broadcast message has to be sent in the sensor network. Nodes may have multiple interfaces, therefore policies has to be used to define which packet is allowed to be sent over which interface. Policies may change according to needs of the applications and users. Since a network can be used by different applications and users at the same time, different policy profiles have to be able to co-exists on the same node. If needed by an application it should be detected if a node changes its location and the routing has to be changed accordingly. It should be investigated how new nodes will attach to the system and take over the responsibilities of destroyed nodes in the tunnel's infrastructure. Basically there are two possibilities for that. Either the new nodes answer to incoming queries or they inform the network about their existence before answering. It should be analysed how to distinguish between different sensor network regions in terms of management. For example, the sensors in the firemen PANs should be merged into the tunnel's infrastructure; however they still should be controlled by firemen PAN controller.

The traditional model of a sensor network as a very dedicated solution, with a collapsed stack and no internal user authentication / authorization, must be changed. In order to allow for more complex use cases with multiple users and applications, each utilizing their own addressing structure, a more layered approach has to be taken. Depending on the topology of the sensor network, different solutions can be conceived. In a hierarchical topology, a more straightforward approach can be taken and more standardized ways can be used in order to create a finer user authentication / authorization. However, also here, new protocols might be needed that can handle for instance multicasting of requests. The relative simplicity of this approach comes at the cost of having a more restrictive topology. If a flat mesh topology is going to be used, we probably need to enforce authentication and authorization all the way down to the node level. On top of this, the authentication can not be done using asymmetric methods. Therefore, with this approach a completely new method has to be invented in order to achieve distributed policy enforcement points in the sensor network.

As part of the scenarios envisaged by RUNES will require or at least benefit from sensor networks moving between different locations, such as a sensor network brought into a tunnel by a fire-truck, the gateways connecting the sensor network to a WAN need to support this kind of mobility. First approaches for solution in this area have been investigated and assigned to certain sensor networking methods. Further work will integrate these different approaches to an integrated mobility solution set, able to support a broad range of scenarios.

As far as the work in Delay Tolerant Networking is concerned, the results obtained so far are promising and we will continue in light of the steps described in section 11.9, both in the implementation part of the algorithm and in the theoretical part of the routing problem for the DTN networks.

APPENDIX

A. Survey of Ad-hoc routing protocols

At the network layer, mesh connectivity for mobile systems is based around the use of IP routing using ***ad-hoc routing protocols***. A mobile ad-hoc network (MANET) can be described as[3]:

A mobile ad-hoc network (MANET) is a self-configuring network of mobile routers (and associated hosts) connected by wireless links—the union of which form an arbitrary topology. The routers are free to move randomly and organise themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet.

Since around the mid-90s, MANETs have attracted a great deal of attention. However, there is still not established core set of protocols for MANET usage even though there is standardisation work within the IETF MANET WG [95]. Although the MANET WG is working on three core protocols [101][102][103], there are more than 70 protocols known of within the research community! A list can be found at [96], with links to documents and some implementations also, and some of the discussion in this section is based on that information.

Rather than present an exhaustive survey of the many, many protocols, this section will:

- Give a description of the many types (categories) of protocols and their salient features.
- Give a summary of the three MANET protocols from the IETF MANET WG.
- Discuss the relevance to our three Mesh Application Scenarios.

C*urrent* sensor networks tend to consist of resource poor wireless sensors and IP based protocols are considered too heavy weight [97][98]. However, as sensors become more capable, more of the MANET techniques will be applied directly to such platforms.

A.1. Categories of MANET protocols

There are several categories into which MANET protocols can be placed based on their modes of operation and the basis of the routing algorithm that is used. The main categories are:

- Proactive (Table-Driven)
- Reactive (On-Demand)
- Hierarchical
- Geographical
- Power Aware (Energy Efficient)
- Multicast (Multipoint)
- Geocast (Geographical Multicast)
- Other

The main protocols that are used are

[3] http://en.wikipedia.org/wiki/Mobile_ad-hoc_network

Proactive: These protocols are closely related to their wired network counterparts in that they build routing and forwarding tables by proactively (periodically, for example) soliciting/transmitting information from/to other network nodes (neighbours or the entire network, depending on protocol). This information is then used to create an updated table of destinations within the network and is used to build the forwarding table used to relay packets. Proactive protocols have the advantage that they will have lower forwarding latency as all routes are computed ahead of use. They have the disadvantage that they expend resources (network capacity and battery power) in discovering routes that may never be used, or routes that change before they are ever used.

Reactive: Reactive protocols build routes on demand: when a node receives a packet for a destination that it does not know about (it does not have in its route cache), it discovers the route and adds it to its route cache. Reactive protocols have three phases in operation: *route discovery*, *packet forwarding* and *route maintenance*. Route discovery is finding a route to a given destination. This is often done through some modified flooding system of discovery packets. When the discovery packet gets to the intended destination, it sends a response to the discovery packet, often using reverse path forward (backward learning) to the source. All nodes on the (reverse) path can then cache the route to that destination. Forwarding can be table driven or use source routing, depending on the protocol. For route maintenance, explicit keep-alive packets and/or packet sniffing (for packets with the destination as a source address) then keep the route "fresh". If the route goes "stale", error packets may be sent upstream to indicate a broken route (or partial repair could be attempted from the point at which the route break is detected). Reactive protocols have the advantage that they keep route information only for routes that are being used. They have the disadvantage of latency in forwarding when discovering a route that is not already known, and also they may generate a lot of traffic for route discovery and route maintenance.

Hierarchical: These protocols split the ad-hoc network into a hierarchy of nodes sometimes called zones. Routing occurs only within a zone, and some nodes from the gateways between zones. We are not aware of much activity on such protocols within the research at the present time.

Geographical: These protocols exploit some knowledge of location or geography in order to make routing decisions. They have the constraint that some form of location or coordinate system must be available to act as reference for the operation of the protocol. We are not aware of much activity on such protocols within the research at the present time.

Power Aware (Energy Efficient): These protocols use algorithms that conserve energy within the network as a whole, not just for individual nodes. This is may be achieved by using some form of energy or power measure as a routing metric and then performing a form of load balancing that prevents routing packets always along the same paths to avoid energy depletion for nodes along that path. There is currently great interest in such protocols, especially in the area of wireless sensor networks.

Multicast and Geocasting: Often extensions to their proactive or reactive unicast cousins, these protocols try to achieve optimised delivery of the same packet to multiple destinations. Geocasting further tries to exploit location information to achieve multipoint delivery. We are not aware of much activity on such protocols within the research at the present time.

Other: There are also protocols based on temporal information and constrained flooding, and protocols that try to optimise for QoS. Relevant protocols will be discussed later in the study as the need arises.

Most research effort has been aimed at proactive and reactive protocols, with a lot of recent activity on Power Aware protocols in the area of wireless sensor networks.

A.2. The IETF MANET WG

A good description of this WG comes from its charter [95], and is given below.

The purpose of the MANET working group is to standardise IP routing protocol functionality suitable for wireless routing application within both static and dynamic topologies with increased dynamics due to node motion or other factors.

Approaches are intended to be relatively lightweight in nature, suitable for multiple hardware and wireless environments, and address scenarios where MANETs are deployed at the edges of an IP infrastructure. Hybrid mesh infrastructures (e.g., a mixture of fixed and mobile routers) should also be supported by MANET specifications and management features.

Using mature components from previous work on experimental reactive and proactive protocols, the WG will develop two Standards track routing protocol specifications:

- Reactive MANET Protocol (RMP)
- Proactive MANET Protocol (PMP)

If significant commonality between RMP and PMP protocol modules is observed, the WG may decide to go with a converged approach. Both IPv4 and IPv6 will be supported. Routing security requirements and issues will also be addressed.

Below, we give a description of the protocol work in progress within this WG. Much of the information below is taken from [101][102][103] and from work-in-progress documented in Internet Drafts that re linked to the main MANET WG WWW page [95].

A.3. Ad-Hoc on Demand Distance Vector Routing (AODV)

The following description of AODV is taken from the IETF RFC3561 [101].

The Ad hoc On-Demand Distance Vector (AODV) algorithm enables dynamic, self-starting, multihop routing between participating mobile nodes wishing to establish and maintain an ad hoc network. AODV allows mobile nodes to obtain routes quickly for new destinations, and does not require nodes to maintain routes to destinations that are not in active communication. AODV allows mobile nodes to respond to link breakages and changes in network topology in a timely manner.

The operation of AODV is loop-free, and by avoiding the Bellman-Ford "counting to infinity" problem offers quick convergence when the ad hoc network topology changes (typically, when a node moves in the network). When links break, AODV causes the affected set of nodes to be notified so that they are able to invalidate the routes using the lost link.

One distinguishing feature of AODV is its use of a destination sequence number for each route entry. The destination sequence number is created by the destination to be included along with any route information it sends to requesting nodes. Using destination sequence numbers ensures loop freedom and is simple to program. Given the choice between two routes to a destination, a requesting node is required to select the one with the greatest sequence number.

A.3.1. Overview

Route Requests (RREQs), Route Replies (RREPs), and Route Errors (RERRs) are the message types defined by AODV. These message types are received via UDP, and normal IP header processing applies. So, for instance, the requesting node is expected to use its IP address as the Originator IP

address for the messages. For broadcast messages, the IP limited broadcast address (255.255.255.255) is used.

This means that such messages are not blindly forwarded. However, AODV operation does require certain messages (e.g., RREQ) to be disseminated widely, perhaps throughout the ad hoc network. The range of dissemination of such RREQs is indicated by the TTL in the IP header. Fragmentation is typically not required.

As long as the endpoints of a communication connection have valid routes to each other, AODV does not play any role. When a route to a new destination is needed, the node broadcasts a RREQ to find a route to the destination. A route can be determined when the RREQ reaches either the destination itself, or an intermediate node with a 'fresh enough' route to the destination. A 'fresh enough' route is a valid route entry for the destination whose associated sequence number is at least as great as that contained in the RREQ. The route is made available by unicasting a RREP back to the origination of the RREQ.

Each node receiving the request caches a route back to the originator of the request, so that the RREP can be unicast from the destination along a path to that originator, or likewise from any intermediate node that is able to satisfy the request.

Nodes monitor the link status of next hops in active routes. When a link break in an active route is detected, a RERR message is used to notify other nodes that the loss of that link has occurred. The RERR message indicates those destinations (possibly subnets) which are no longer reachable by way of the broken link. In order to enable this reporting mechanism, each node keeps a "precursor list", containing the IP address for each its neighbours that are likely to use it as a next hop towards each destination. The information in the precursor lists is most easily acquired during the processing for generation of a RREP message, which by definition has to be sent to a node in a precursor list. If the RREP has a nonzero prefix length, then the originator of the RREQ which solicited the RREP information is included among the precursors for the subnet route (not specifically for the particular destination).

A RREQ may also be received for a multicast IP address. In this document, full processing for such messages is not specified. For example, the originator of such a RREQ for a multicast IP address may have to follow special rules. However, it is important to enable correct multicast operation by intermediate nodes that are not enabled as originating or destination nodes for IP multicast addresses, and likewise are not equipped for any special multicast protocol processing. For such multicast-unaware nodes, processing for a multicast IP address as a destination IP address MUST be carried out in the same way as for any other destination IP address.

AODV is a routing protocol, and it deals with route table management. Route table information must be kept even for short-lived routes, such as are created to temporarily store reverse paths towards nodes originating RREQs. AODV uses the following fields with each route table entry:

- Destination IP Address
- Destination Sequence Number
- Valid Destination Sequence Number flag
- Other state and routing flags (e.g., valid, invalid, repairable, being repaired)
- Network Interface
- Hop Count (number of hops needed to reach destination)
- Next Hop
- List of Precursors

- Lifetime (expiration or deletion time of the route)

Managing the sequence number is crucial to avoiding routing loops, even when links break and a node is no longer reachable to supply its own information about its sequence number. A destination becomes unreachable when a link breaks or is deactivated. When these conditions occur, the route is invalidated by operations involving the sequence number and marking the route table entry state as invalid.

A.4. Optimized Link State Routing Protocol (OLSR)

The following description of OLSR is taken from the IETF RFC RFC3626 [102].

The Optimized Link State Routing Protocol (OLSR) is developed for mobile ad hoc networks. It operates as a table driven, proactive protocol, i.e., exchanges topology information with other nodes on the network regularly. Each node selects a set of its neighbour nodes as "multipoint relays" (MPR). In OLSR, only nodes, selected as such MPRs, are responsible for forwarding control traffic, intended for diffusion into the entire network. MPRs provide an efficient mechanism for flooding control traffic by reducing the number of transmissions required.

Nodes, selected as MPRs, also have a special responsibility when declaring link state information in the network. Indeed, the only requirement for OLSR to provide shortest path routes to all destinations is that MPR nodes declare link-state information for their MPR selectors. Additional available link-state information may be utilized, e.g., for redundancy.

Nodes which have been selected as multipoint relays by some neighbour node(s) announce this information periodically in their control messages. Thereby a node announces to the network, that it has reachability to the nodes which have selected it as an MPR. In route calculation, the MPRs are used to form the route from a given node to any destination in the network. Furthermore, the protocol uses the MPRs to facilitate efficient flooding of control messages in the network.

A node selects MPRs from among its one hop neighbours with "symmetric", i.e., bi-directional, linkages. Therefore, selecting the route through MPRs automatically avoids the problems associated with data packet transfer over uni-directional links (such as the problem of not getting link-layer acknowledgments for data packets at each hop, for link-layers employing this technique for unicast traffic).

OLSR is developed to work independently from other protocols. Likewise, OLSR makes no assumptions about the underlying link-layer.

A.4.1. Overview

OLSR is a proactive routing protocol for mobile ad hoc networks. The protocol inherits the stability of a link state algorithm and has the advantage of having routes immediately available when needed due to its proactive nature. OLSR is an optimization over the classical link state protocol, tailored for mobile ad hoc networks.

OLSR minimizes the overhead from flooding of control traffic by using only selected nodes, called MPRs, to retransmit control messages. This technique significantly reduces the number of retransmissions required to flood a message to all nodes in the network. Secondly, OLSR requires only partial link state to be flooded in order to provide shortest path routes. The minimal set of link state information required is, that all nodes, selected as MPRs, MUST declare the links to their MPR selectors. Additional topological information, if present, MAY be utilized e.g., for redundancy purposes.

OLSR MAY optimize the reactivity to topological changes by reducing the maximum time interval for periodic control message transmission. Furthermore, as OLSR continuously maintains routes to all destinations in the network, the protocol is beneficial for traffic patterns where a large subset of nodes are communicating with another large subset of nodes, and where the [source, destination] pairs are changing over time. The protocol is particularly suited for large and dense networks, as the optimization done using MPRs works well in this context. The larger and more dense a network, the more optimization can be achieved as compared to the classic link state algorithm.

OLSR is designed to work in a completely distributed manner and does not depend on any central entity. The protocol does NOT REQUIRE reliable transmission of control messages: each node sends control messages periodically, and can therefore sustain a reasonable loss of some such messages. Such losses occur frequently in radio networks due to collisions or other transmission problems.

Also, OLSR does not require sequenced delivery of messages. Each control message contains a sequence number which is incremented for each message. Thus the recipient of a control message can, if required, easily identify which information is more recent - even if messages have been re-ordered while in transmission.

Furthermore, OLSR provides support for protocol extensions such as sleep mode operation, multicast-routing etc. Such extensions may be introduced as additions to the protocol without breaking backwards compatibility with earlier versions.

OLSR does not require any changes to the format of IP packets. Thus any existing IP stack can be used as is: the protocol only interacts with routing table management.

A.5. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)

The following description of TBRPF is taken from the IETF RFC3684 [103].

Topology Dissemination Based on Reverse-Path Forwarding (TBRPF) is a proactive, link-state routing protocol designed for mobile ad-hoc networks (MANETs), which provides hop-by-hop routing along shortest paths to each destination. Each node running TBRPF computes a source tree (providing shortest paths to all reachable nodes) based on partial topology information stored in its topology table, using a modification of Dijkstra's algorithm. To minimize overhead, each node reports only *part* of its source tree to neighbours.

TBRPF uses a combination of periodic and differential updates to keep all neighbours informed of the reported part of its source tree. Each node also has the option to report addition topology information (up to the full topology), to provide improved robustness in highly mobile networks.

TBRPF performs neighbour discovery using "differential" HELLO messages which report only *changes* in the status of neighbours. This results in HELLO messages that are much smaller than those of other link-state routing protocols such as OSPF.

A.5.1. Overview

The TBRPF Neighbour Discovery (TND) protocol allows each node i to quickly detect the neighbour nodes j such that a bidirectional link (I,J) exists between an interface I of node i and an interface J of node j. The protocol also quickly detects when a bidirectional link breaks or becomes unidirectional.

The key feature of TND is that it uses "differential" HELLO messages which report only *changes* in the status of links. This results in HELLO messages that are much smaller than those of other link-

state routing protocols such as OSPF, in which each HELLO message includes the IDs of *all* neighbours. As a result, HELLO messages can be sent more frequently, which allows faster detection of topology changes.

TND is designed to be fully modular and independent of the routing module. TND performs ONLY neighbour sensing, i.e., it determines which nodes are (1-hop) neighbours. In particular, it does not discover 2-hop neighbours (which is handled by the routing module). As a result, TND can be used by other routing protocols, and TBRPF can use another neighbour discovery protocol in place of TND, e.g., one provided by the link layer.

Nodes with multiple interfaces run TND separately on each interface, similar to OSPF. Thus, a neighbour table is maintained for each local interface, and a HELLO sent on a particular interface contains only information regarding neighbours heard on that interface.

We note that, in wireless networks, it is possible for a single interface I to receive packets from multiple interfaces J associated with the same neighbour node. This could happen, for example, if the neighbour uses a directional antenna with different interfaces representing different beams. For this reason, TBRPF includes neighbour interface addresses in HELLO messages, unlike OSPF, which includes only router IDs in HELLO packets.

Each TBRPF node maintains a neighbour table for each local interface I, which stores state information for each neighbour interface J heard on that interface, i.e., for each link (I,J) between interface I and a neighbour interface J. The status of each link can be 1-WAY, 2-WAY, or LOST. The neighbour table for interface I determines the contents of HELLO messages sent on interface I, and is updated based on HELLO messages received on interface I (and possibly on link-layer notifications).

Each TBRPF node sends (on each interface) at least one HELLO message per HELLO_INTERVAL. Each HELLO message contains three (possibly empty) lists of neighbour interface addresses (which are formatted as three message subtypes): NEIGHBOUR REQUEST, NEIGHBOUR REPLY, and NEIGHBOUR LOST. Each HELLO message also contains the current HELLO sequence number (HSEQ), which is incremented with each transmitted HELLO.

In the following overview of the operation of TND, we assume that interface I belongs to node i, and interface J belongs to node j. When a node i changes the status of a link (I,J), it includes the neighbour interface address J in the appropriate list (NEIGHBOUR REQUEST/REPLY/LOST) in at most NBR_HOLD_COUNT (typically 3) consecutive HELLOs sent on interface I. This ensures that node j will either receive one of these HELLOs on interface J, or will miss NBR_HOLD_COUNT HELLOs and thus declare the link (J,I) to be LOST. This technique makes it unnecessary for a node to include each 1-WAY or 2-WAY neighbour in HELLOs indefinitely, unlike OSPF.

To avoid establishing a link that is likely to be short lived (i.e., to employ hysteresis), node i must receive (on interface I) at least HELLO_ACQUIRE_COUNT (e.g., 2) of the last HELLO_ACQUIRE_WINDOW (e.g., 3) HELLOs sent from a neighbour interface J, before declaring the link (I,J) to be 1-WAY. When this happens, node i includes J in the NEIGHBOUR REQUEST list in each of its next NBR_HOLD_COUNT HELLO messages sent on interface I, or until a NEIGHBOUR REPLY message containing I is received on interface I from neighbour interface J.

If node j receives (on interface J) one of the HELLOs sent from interface I that contains J in the NEIGHBOUR REQUEST list, then node j declares the link (J,I) to be 2-WAY (unless it is already 2-WAY), and includes I in the NEIGHBOUR REPLY list in each of its next NBR_HOLD_COUNT HELLO messages sent on interface J. Upon receiving one of these HELLOs on interface I, node i declares the link (I,J) to be 2-WAY.

If node i receives a HELLO on interface I, sent from neighbour interface J, whose HSEQ indicates that at least NBR_HOLD_COUNT HELLOs were missed, or if node i receives no HELLO on interface I sent from interface J within NBR_HOLD_TIME seconds, then node i changes the status of link (I,J) to LOST (unless it is already LOST), and includes J in the NEIGHBOR LOST list in each of its next NBR_HOLD_COUNT HELLO messages sent on interface I (unless the link changes status before these transmissions are complete). Node j will either receive one of these HELLOs on interface J or will miss NBR_HOLD_COUNT HELLOs; in either case, node j will declare the link (J,I) to be LOST. In this manner, both nodes will agree that the link between I and J is no longer bidirectional, even if node j can still hear HELLOs from node i.

Each node may maintain and update one or more link metrics for each link (I,J) from a local interface I to a neighbour interface J, representing the quality of the link. Such link metrics can be used as additional conditions for changing the status of a neighbour, based on the link metric going above or below some threshold. TBRPF also allows link metrics to be advertised in topology updates, and to be used for computing shortest paths.

A.6. Dynamic Source Routing Protocol (DSR)

The following description of DSR is taken from the current work-in-progress[4] DSR Internet Draft on [95]

The Dynamic Source Routing protocol (DSR) is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. DSR allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. The protocol is composed of the two main mechanisms of "Route Discovery" and "Route Maintenance", which work together to allow nodes to discover and maintain routes to arbitrary destinations in the ad hoc network. All aspects of the protocol operate entirely on-demand, allowing the routing packet overhead of DSR to scale automatically to only that needed to react to changes in the routes currently in use. The protocol allows multiple routes to any destination and allows each sender to select and control the routes used in routing its packets, for example for use in load balancing or for increased robustness. Other advantages of the DSR protocol include easily guaranteed loop-free routing, operation in networks containing unidirectional links, use of only "soft state" in routing, and very rapid recovery when routes in the network change. The DSR protocol is designed mainly for mobile ad hoc networks of up to about two hundred nodes, and is designed to work well with even very high rates of mobility.

A.6.1. Overview

The DSR protocol is composed of two main mechanisms that work together to allow the discovery and maintenance of source routes in the ad hoc network:

- Route Discovery is the mechanism by which a node S wishing to send a packet to a destination node D obtains a source route to D. Route Discovery is used only when S attempts to send a packet to D and does not already know a route to D.
- Route Maintenance is the mechanism by which node S is able to detect, while using a source route to D, if the network topology has changed such that it can no longer use its route to D because a link along the route no longer works. When Route Maintenance indicates a source route is broken, S can attempt to use any other route it happens to know to D, or can invoke

[4] http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt at time of writing

Route Discovery again to find a new route for subsequent packets to D. Route Maintenance for this route is used only when S is actually sending packets to D.

In DSR, Route Discovery and Route Maintenance each operate entirely "on demand". In particular, unlike other protocols, DSR requires no periodic packets of any kind at any layer within the network. For example, DSR does not use any periodic routing advertisement, link status sensing, or neighbour detection packets, and does not rely on these functions from any underlying protocols in the network. This entirely on-demand behaviour and lack of periodic activity allows the number of overhead packets caused by DSR to scale all the way down to zero, when all nodes are approximately stationary with respect to each other and all routes needed for current communication have already been discovered. As nodes begin to move more or as communication patterns change, the routing packet overhead of DSR automatically scales to only that needed to track the routes currently in use. Network topology changes not affecting routes currently in use are ignored and do not cause reaction from the protocol.

All state maintained by DSR is "soft state", in that the loss of any state will not interfere with the correct operation of the protocol; all state is discovered as needed and can easily and quickly be rediscovered if needed after a failure without significant impact on the protocol. This use of only soft state allows the routing protocol to be very robust to problems such as dropped or delayed routing packets or node failures. In particular, a node in DSR that fails and reboots can easily rejoin the network immediately after rebooting; if the failed node was involved in forwarding packets for other nodes as an intermediate hop along one or more routes, it can also resume this forwarding quickly after rebooting, with no or minimal interruption to the routing protocol.

In response to a single Route Discovery (as well as through routing information from other packets overheard), a node may learn and cache multiple routes to any destination. This support for multiple routes allows the reaction to routing changes to be much more rapid, since a node with multiple routes to a destination can try another cached route if the one it has been using should fail. This caching of multiple routes also avoids the overhead of needing to perform a new Route Discovery each time a route in use breaks. The sender of a packet selects and controls the route used for its own packets, which together with support for multiple routes also allows features such as load balancing to be defined. In addition, all routes used are easily guaranteed to be loop-free, since the sender can avoid duplicate hops in the routes selected.

The operation of both Route Discovery and Route Maintenance in DSR are designed to allow unidirectional links and asymmetric routes to be supported. In particular, in wireless networks, it is possible that a link between two nodes may not work equally well in both directions, due to differing antenna or propagation patterns or sources of interference.

A.7. Dynamic MANET On-demand (DYMO) Routing

The following description of DYMO is taken from the current work in progress[5] DYMO Internet Draft on [95].

The Dynamic MANET On-demand (DYMO) routing protocol is intended for use by mobile nodes in wireless multihop networks. It offers adaptation to changing network topology and determines unicast routes between nodes within the network.

[5] http://www.ietf.org/internet-drafts/draft-ietf-manet-dymo-01.txt at time of writing

A.7.1. Overview

The Dynamic MANET On-demand (DYMO) routing protocol enables reactive, multihop routing between participating nodes that wish to communicate. The basic operations of the DYMO protocol are route discovery and management. During route discovery the originating node initiates dissemination of a Route Request (RREQ) throughout the network to find the target node. During this dissemination process, each intermediate node records a route to the originating node. When the target node receives the RREQ, it responds with a Route Reply (RREP), unicast toward the originating node. Each node that receives the RREP records a route to the target node, and then the RREP is unicast toward the originating node. When the originating node receives the RREP, routes have then been established between the originating node and the target node in both directions.

In order to react to changes in the network topology nodes maintain their routes and monitor their links. When a packet is received for a route that is no longer available the source of the packet is notified. A Route Error (RERR) is sent to the packet source to indicate the current route is broken. Once the source receives the RERR, it re-initiates route discovery if it still has packets to deliver.

In order to enable extension of the base specification, DYMO defines a generic element structure and handling of future extensions. By defining a fixed structure and default handling, future extensions are handled in a predetermined fashion.

DYMO uses sequence numbers as they have been proven to ensure loop freedom. Sequence numbers enable nodes to determine the order of DYMO route discovery packets, thereby avoiding use of stale routing information.

All DYMO packets are transmitted via UDP.

A.8. Ad Hoc routing for the Mesh Application Scenarios

Intuitively, one might consider that the reactive protocols are better suited to higher mobility rates and indeed this is the finding of an early study [99], though there have not been any more recent comparative performance studies. However, another study from around the same time [100] says that although DSR (reactive) can perform better that AODV (proactive), it can be heavily dependent on the rate of mobility (i.e. the application scenario) but also on the tuning of the protocol (e.g. the time-out periods for DSR route cache entries). Performance analysis of wireless ad hoc networks is a must take into account a number of interactions across the network architecture layers: the interactions between the wireless physical layer, radio propagation, multiple access, random and changing topology due to mobility, routing, and the characteristics of the application that generates the traffic carried by the network.

So, it is not easy to make prescriptive recommendations for the applicability of specific ad hoc routing algorithms and protocols within the mesh scenarios discussed here. Indeed, it is something that currently has a high level of research activity and it is quite possible that certain other non-functional requirements might have a greater bearing on the choice of protocol than the functional capability of the protocols.

References

[1] Domonkos Asztalos, Karen Lawson, Stephen Hailes, Lesley Hanna, Ingolf Krüger: "Application Scenario Building/Definition", IST RUNES Deliverable D2.1, http://www.ist-runes.org/docs/deliverables/D2_01.pdf, March 31st 2005

[2] IST Ambient Netwroks: http://www.ambient-networks.org

[3] IEEE, "Guidelines for 64 bit global identifier (EUI-64) Registration Authority", http://standards.ieee.org/regauth/oui/tutorials/EUI64.html

[4] IEEE Standard 802.15.4-2003, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)", May 2003

[5] G. Montenegro and N. Kushalnagar, " Transmission of IPv6 Packets over IEEE 802.15.4 Networks ", draft-ietf-6lowpan-format-01, Internet-Draft, work in progress, October 2005

[6] http://www.ietf.org/html.charters/6lowpan-charter.html

[7] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, December 1998.

[8] J. Jeong, J. Park, H. Kim, H. Jeong, D. Kim, "Ad Hoc IP Address Autoconfiguration", draft-jeong-adhoc-ip-addr-autoconf-05.txt, Internet Draft (Work in Progress), July 2005

[9] N. Kim, Y. Lee, S. Ahn, AROD: An address autoconfiguration with Address Reservation and Optimistic Duplicated address detection for mobile ad hoc networks", draft-nhkim-arod-autoconf-00.txt, Internet Draft (Work in Progress), November 2005

[10] S. Singh, J. Kim, C. Perkins, T. Clausen, P. Ruiz, " Ad hoc network autoconfiguration: definition and problem statement", draft-singh-autoconf-adp-02.txt, Internet Draft (Work in Progress), October 2005

[11] Ryuji Wakikawa, Jari T. Malinen, Charles E. Perkins, Anders Nilsson, Antti J. Tuominen, "Global connectivity for IPv6 Mobile Ad Hoc Networks", draft-wakikawa-manet-globalv6-04.txt, Internet Draft (Work in Progress), July 2005

[12] F. Ros, P. Ruiz, C. Perkins, "Extensible MANET Auto-configuration Protocol (EMAP)", draft-ros-autoconf-emap-01.txt, Internet Draft (Work in Progress), October 2005

[13] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", IETF RFC 4193, October 2005.

[14] J. Macker, et al, "Simplified Multicast Forwarding for MANET", draft-ietf-manet-smf-01.txt, Internet Draft (work in progress), June 2005

[15] T. Clausen, P. Jacquet, "Optimized Link State Routing Protocol (OLSR)", IETF RFC 3626, October 2003

[16] I. Chakeres, E. Belding-Royer, C. Perkins, "Dynamic MANET On-demand (DYMO) Routing", draft-ietf-manet-dymo-03.txt, Internet Draft (work in progress), October 2005

[17] S. Chakrabarti, E. Nordmark, "LowPan Neighbor Discovery Extensions", draft-chakrabarti-lowpan-ipv6-nd-00.txt, Internet Draft (Work in Progress), July 2005

[18] T. Narten, E. Nordmark, W. Simpson" Neighbor Discovery for IP Version 6 (IPv6)", IETF RFC 2461, December 1998

[19] S. Thomson, T. Narten, " IPv6 Stateless Address Autoconfiguration", IETF RFC 2462, December 1998

[20] Karl Mayer, Björn Grönvall, Janne Riihijärvi, Annikki Welin, Gösta Leijonhufvud, Frank Oldewurtel, Wolfgang Fritsche, Mattias Johansson, Stephen Hailes, Cecilia Mascolo, Mirco Musolesi, Håkan Hjalmarsson: *"Requirements and Configuration Scenarios"*, RUNES Deliverable 4.1, April 2005

[21] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, S. Singh: *"Exploiting Heterogeneity in Sensor Networks"*, Infocom 2005, March 2005

[22] R. Govindan, E. Kohler, D. Estrin, F. Bian, K. Chintalapudi, O. Gnawali, R. Gummadi, S. Rangwala, T. Stathopoulos: "*Tenet: An Architecture for Tiered Embedded Networks"*, ?

[23] A. Koubaa, M. Alves, E. Tovar: "*A Real Time Sensor Network Architecture with Power-Efficient Resource Reservation in IEEE 802.15.4"*, Techincal Report, August 2005

[24] S. Jain, K. Fall, R. Patra, "Routing in a Delay Tolerant Networking", SIGCOMM, Aug/Sep 2004.

[25] B. Kotnyek. An Annotated Overview of Dynamic Network Flows. Technical Report RR-4936, INRIA, Sept. 2003.

[26] L. R. Ford and D. R. Fulkerson. Flows in Networks. Princeton University Press, 1962.

[27] A. Orda and R. Rom. Shortest-Path and Minimum Delay Algorithms in Networks with Time-Dependent Edge-Length. Journal of the ACM, 37(3), 1990.

[28] R. Ogier. Minimum-delay Routing in Continuous-time Dynamic Networks with Piecewise-constant Capacities. Computer Networks, 18:303-318, 1988.

[29] X. Chen and A. L. Murphy. Enabling Disconnected Transitive Communication in Mobile Adhoc Networks. In Workshop on Principles of Mobile Computing, August 2001.

[30] P. Juang, H. Oki, Y. Wang, M. Margaret, P. Li-Shiuan, and R. Daniel. Energy-E_cient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In ASPLOS-X, October 2002.

[31] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks. In IEEE SNPA, May 2003.

[32] A. Vahdat and D. Becker. Epidemic Routing for Partially-connected Ad hoc Networks. Technical Report CS-2000-06, Duke University, July 2000.

[33] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, andJ. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In ACM Mobicom, Aug. 1998.

[34] A. Lindgren, A. Doria, O. Schelen, "Probabilistic Routing in Intermittently Connected Networks", In Proceedings of the The First International Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR 2004), August 2004, Fortaleza, Brazil.

[35] K.Langendoen and N.Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison." Compu. Networks, 43(4): 499-518, 2003

[36] Lance Doherty, Kristofer Pister and Laurent El Ghaoui. Convex Position Estimation in Wireless Sensor Networks. *IEEE InfoCom 2001*. April 2001.)

[37] Yi Shang, Wheeler Ruml, Ying Zhang, Markus Fromherz. Localization from Mere Connectivity. *MobiHoc 2003*. June 2003

[38] Yi Shang and Wheeler Ruml, "Improved MDS Based Localization", in Proceedings of IEEE INFOCOM 2004, April, 2004

[39] B.H.Wellenhoff, H.Lichtenegger and J. Collins, Global Positioning System: Theory and Practice, Fourth Edition, Springer Verlag, 1997

[40] Andreas Savvides, Chih-Chieh Han, Mani B. Strivastava. Dynamic fine-grained localization in Ad-Hoc networks of sensors. MobiCom 2001, Rome, Italy, July 2001

[41] N. B. Priyantha, A. Chakraborty and H. Balakrishnan, The Cricket Location-Support System, In Proceedings of MOBICOM '00, New York, August 2000.

[42] D. Niculescu and B. Nath, Ad Hoc Positioning System (APS) using AoA, *INFOCOM' 03*, San Francisco, CA, 2003

[43] P. Bahl and V. N. Padmanabhan, RADAR: An In-Building RF-Based User Location and Tracking System, In Proceedings of the IEEE INFOCOM '00, March 2000.

[44] J. Hightower, G. Boriello and R. Want, SpotON: An indoor 3D Location Sensing Technology Based on RF Signal Strength, University of Washington CSE Report #2000-02-02, February 2000.

[45] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin and S. Wicker, Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks, Technical Report UCLA/CSD-TR 02-0013, 2002.

[46] L. Girod and D. Estrin, Robust Range Estimation using Acoustic and Multimodal Sensing, In Proceedings of IROS'01, Maui, Hawaii, October 2001.

[47] C. Savarese, J. Rabay and K. Langendoen, Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks, USENIX Technical Annual Conference, Monterey, CA, June 2002.

[48] A. Savvides, H. Park and M. Srivastava, The Bits and Flops of the N-Hop Multilateration Primitive for Node Localization Problems, In First ACM International *Workshop on Wireless Sensor Networks and Application*, Atlanta, GA, September 2002.

[49] K. Whitehouse and D. Culler, Calibration as Parameter Estimation in Sensor Networks, In *First ACM International Workshop on Wireless Sensor Networks and Application*, Atlanta GA, September 2002.

[50] Dragos Niculescu and Badri Nath, "Ad Hoc Positioning System (APS)", GLOBECOM 2001

[51] C. Randell, et al., "Personal Position Measurement Using Dead Reckoning," *Proc. 2003 ISWC.*

[52] C. T. Judd, "A Personal Dead Reckoning Module," Proc. ION GPS '97. Online at http://www1.cs.columbia.edu/~drexel/CandExam/DRM_ION97paper.pdf

[53] http://www.pointresearch.com/drm3_module.htm

[54] FRID-Assisted Localization and Communication for First Responders, NIST interim Project Report of Advance Network Technologies Division, on line at http://www.pointresearch.com/drm3_module.htm

[55] Nirupama Bulusu, John Heidemann and Deborah Estrin. GPS-less Low Cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications Magazine*. October 2000.

[56] Nirupama Bulusu, John Heidemann and Deborah Estrin. Density Adaptive Algorithms for Beacon Placement in Wireless Sensor Networks. *IEEE ICDCS 2001*. April 2001.

[57] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, Tarek Abdelzaher. Range-free Localization Schemes for Large Scale Sensor Networks. *MobiCom 2003*.

[58] Dragos Niculescu and Badri Nath. DV Based Positioning in Ad hoc Networks. *Kluwer Journal of Telecommunication Systems*. 2003.

[59] Radhika Nagpal, Howard Shrobe, and Jonathan Bachrach. "Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network", in *Proceedings of 2nd International Workshop on Information Processing in Sensor Networks (IPSN)*. April 2003.

[60] "A Data Fusion Architecture for Enhanced position Estimation in Wireless Networks", IEEE Communication Letters

[61] Lingxuan Hu et al, "Localization for Mobile Sensor Networks", MobiCom'04

[62] Jeffrey Hightower and Gaerano Borriello, "Location Systems for Ubiquitous Computing", IEEE Computer Magazine Aug 2001

[63] CCITT. The director: Overview of concept, models and service. Recommendation X.500, CCITT, 1988.

[64] W. Yeong, T. Howes, and S. Kille. Lightweight directory access protocol. RFC 1777, Internet RFC, March 1995.

[65] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system", Proceedings of the 17th Symposium on Operating Systems Principles, pp.: 186–201, Kiawah Island, SC, USA, December 1999.

[66] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming", Proceedings of the Symposium on Operating Systems Principles, pp.: 146–159, October 2001.

[67] Y.-B. Ko, and N. H. Vaidya, "Geocasting in mobile ad hoc networks: location-based multicast algorithms", Mobile Computing Systems and Applications (WMCSA), February 1999, pp.:101-110

[68] Y. Ko, and N.H. Vaidya, “GeoTORA: A protocol for Geocasting in mobile ad hoc networks”, Conference on Networking Protocols (ICNP), November 2000

[69] W.-H. Liao, Y.-C- Tseng, K.-L. Lo, and J.-P. Sheu, “Geogrid: A Geocasting protocol for mobile ad hoc networks based on grid”, Journal of Internet Technology, Vol. 1, No. 2, pp.: 23-32, 2000

[70] J. C. Navas, and T. Imielinski, “Geocast - geographic addressing and routing”, Mobile Computing and Networking (MobiCom), September 1997

[71] I. Stojmenovic, “Geocasting with guaranteed delivery in sensor networks”, Wireless Communications, Vol. 11, No. 6, December 2004, pp.: 29-37

[72] K. Seada, and A. Helmy, ”Efficient geocasting with perfect delivery in wireless networks”, Wireless Communications and Networking Conference (WCNC), Vol. 4, March 2004, pp.: 2551-2556

[73] Johnson, D., Perkins, C., Arkko, J., "Mobility Support in IPv6", RFC 3775, Januar 2004.

[74] Devarapalli, V., Wakikawa, R., Petrescu, A., Thubert, P., "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, Januar 2005.

[75] RUNES WP 4, "Requirements and Configuration Scenrios", deliverable D4.1, April 2005.

[76] Soliman, H., Tsitsirtis, G., Devarapalli, V., Kempf, J., Levkowetz, H., Thubert, P., Wakikawa, R., " Dual Stack Mobile IPv6 (DSMIPv6) for Hosts and Routers", draft-ietf-mip6-nemo-v4traversal-00.txt (Work in Progress), October 2005

[77] M. Lad, S. Bhatti, S. Hailes, P. Kirstein. "Enabling Coalition-Based Community Networking". The London Communications Symposium 2005 (LCS 2005), 8-9 September 2005.

[78] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771 (Draft Standard), March 1995.

[79] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica, “A Unifying Link Abstraction for Wireless Sensor Networks”, SenSys ‘05

[80] G. Montenegro and N. Kushalnagar, “Transmission pf IPv6 Packets over IEEE 802.15.4 Networks”, internet draft, work in progress, 2005

[81] GOLLUM WP2, “Generic Open Link-Layer API for Unified Media Access”, deliverable D2.1, December 2004.

[82] S. Corson, J. Macker, “Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations”, IETF RFC2501, Jan 1999. http://www.ietf.org/rfc/rfc2501.txt

[83] S. Guha, R. Murty and E. Sirer, “Sextant: A Unified Node and Event Localization Framework using Non-Convex Constraints”, MobiHoc’05, May 2005.

[84] M. Arulampalam, S. maskell, N. Gordon and T. Clapp, “A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian bayesian Tracking”, IEEE Trans. On Signaling Processing, Vol.50, No.2, Feb. 2002

[85] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson and P. Nordlund, "Particle Filters for Positioning, Navigation, and Tracking", IEEE Trans. On Signaling Processing, Vol.50, No.2 Feb. 2002

[86] D. Fox, J. Hightower, L. Liao, D. Schulz and G. Borriello, " Bayesian Filters for Location Estimation", IEEE Pervasive Computing, Sept. 2003

[87] J. VerMaak, S. Godsill and P. Perez, "Monte Carlo Filtering for Multo-Target Tracking and Data Association", IEEE Trans. On Aerospace and Electronic Systems, Vol. 41, No.1 Jan. 2005

[88] Jeff Boleng: "*Efficient Network Layer Addressing for Mobile Ad Hoc Networks*", in Proc. of International Conference on Wireless Networks (ICWN02), Las Vegas, USA, 2002

[89] Kilian A. Weniger: "*Passive Duplicate Address Detection in Mobile Ad Hoc Networks*", in Proc. of IEEE WCNC 2003, New Orleans, USA, March 2003.

[90] Kilian A. Weniger: "*Passive Auto-configuration of Mobile Ad hoc Networks*", Telematics Technical Reports TM-2004-1, ISSN 1613-849X, April 2004

[91] Nitin H. Vaidya: "*Weak Duplicate Address Detection in Mobile Ad Hoc Networks*", MOBIHOC02, EPFL Lausanne, Switzerland, 2002

[92] Y. Peiling, E. Krohne, and T. Camp, "Performance comparison of geocast routing protocols for a MANET", Computer Communications and Networks (ICCCN), 2004, pp.: 213-220

[93] http://www.ietf.org/html.charters/6lowpan-charter.html

[94] A. Savvides, H. Park and M. Srivastava, The Bits and Flops of the N-Hop Multilateration Primitive for Node Localization Problems, In First ACM International Workshop on Wireless Sensor Networks and Application, Atlanta, GA, September 2002.

[95] IETF Mobile Ad-Hoc Network WG. http://www.ietf.org/html.charters/manet-charter.html

[96] Wikipedia *Ad Hoc Protocol List* . http://en.wikipedia.org/wiki/Ad_hoc_protocol_list

[97] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "A Survey on Sensor Networks", *IEEE Communications*, Aug 2002, pp102-114

[98] C.Y. Chong; S. P. Kumar, "Sensor networks: Evolution, opportunities, and challenges", Proc IEEE, Aug 2003

[99] Sung-Ju Lee, Mario Gerla, Chai-Keong Toh, "A Simulation Study of Table-Driven and On-Demand Routing Protocols for Mobile Ad Hoc Networks", IEEE Network, Jul/Aug 1999, pp 48-54

[100] S Das, C. Perkins, E. Royer, "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks", Proc. Infocom 200, Mar 2000.

[101] C. Perkins, E. Belding-Royer, S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. IETF RFC3561. July 2003. http://www.ietf.org/rfc/rfc3561.txt

[102] T. Clausen, Ed., P. Jacquet, Ed.. Optimized Link State Routing Protocol (OLSR). IETF RFC3626 October 2003. http://www.ietf.org/rfc/rfc3626.txt

[103] R. Ogier, F. Templin, M. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). IETF RFC3684. February 2004. http://www.ietf.org/rfc/rfc3684.txt

[104] DTN Simulator. Available from http://www.cs.washington.edu/homes/sushjain/pubs2/currentr.html

[105] Leonidas Lymberopoulos. An Adaptive Policy Based Framework for Network Management. PhD Thesis, Imperial College London, July 2004. Available from http://www.doc.ic.ac.uk/~mss

[106] G. Dini and I. M. Savino, "Scalable and Secure Group Rekeying in Wireless Sensor Networks," *IASTED International Conference on Parallel and Distributed Computing and Networks* (*PDCN* 2006), February 14 – 16, 2006, Innsbruck, Austria.

[107] G. Dini and I. M. Savino, "S2RP: a Scalable and Secure Rekeing Protocol for Wireless Sensor Networks," *Technical Report*, Dept. Of Ingegneria dell'Informazione, University of Pisa, December 2005.

[108] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, 24(11):770–772, November 1981.

[109] National Institute of Standards and Technology, *FIPS PUB* 180-1: *Secure Hash Standard*. National Institute for Standards and Technology, Gaithersburg, MD, USA, Apr. 1995.

[110] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security suite for sensor networks," In *Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking* (*MOBICOM*-01), pages 189–199, New York, July 16–21 2001. ACM Press.

[111] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys*, 35(3):309–329, September 2003.

[112] R. Rivest, "The MD5 message-digest algorithm," *Internet Request for Comment RFC* 1321, Internet Engineering Task Force, Apr. 1992.

[113] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, 21(2):120, Feb. 1978.

[114] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey Framework: Versatile Group Key Management," *IEEE Journal on Selected Areas of Communications* (Special Issue on Middleware), 17(9):1614–1631, August 1999.

[115] C. K. Wong, M. G. Gouda, and S. S. Lam, "Secure Group Communications using Key Graphs," *IEEE/ACM Transactions on Networking*, 8(1):16–30, February 2000.

[116] D. Wallner, E. Harder, and R. Agee, "Key Management for Multicast: Issues and Architectures", RFC2627, IETF, 1999.

[117] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and Benny Pinkas, "Multicast Security: A Taxonomy and Some Efficient Constructions", in Proceedings of INFOCOMM, 1999.

[118] Adam Dunkels, Thiemo Voigt, Juan Alonso, and Hartmut Ritter. Distributed TCP Caching for Wireless Sensor Networks. In Proceedings of the Third Annual Mediterranean Ad Hoc Networking Workshop, June 2004.