

An investigation of placement strategies for distributed complex event processing in mobile ad-hoc networks

Bigirimana Fabrice
Master's Thesis Autumn 2013



An investigation of placement strategies for distributed complex event processing in mobile ad-hoc networks

Bigirimana Fabrice

8th November 2013

Abstract

In the last decade sensor networks and complex event processing have been used to enable powerful real world aware applications. Due to energy constraints in sensor networks, distributed complex event processing has been used as a technique to minimize data transmission and save energy.

Mobile adhoc networking has been used to enable applications that use sensors and complex event processing technologies in areas where there are no network communication infrastructures. Similarly to sensor networks, Mobile adhoc networks are characterised by energy constraints, thus distributed complex event processing is preferable in order to limit energy consumption. However, placement strategies used to enable distributed complex event processing in sensor networks are not suitable in mobile adhoc networks due to the dynamic topology.

In this thesis, we investigate placement strategies for distributed complex event processing in mobile adhoc networks. We claim that distributed placement strategies can achieve better distributed complex event processing performance compared to centralized approaches. Therefore, as part of this investigation, we design and implement a distributed placement strategy which we later evaluate in comparison to existing centralized approaches.

Through literature work we identify the main challenges, issues and requirements for complex event processing, sensor data processing and mobile adhoc networking. This is later used as the foundation for designing and implementing the distributed placement mechanism. Due to the volatile nature of mobile computing, our mechanism uses a heuristic approach technique in order to find a near-optimal execution plan for distributed complex event processing.

We use complex event processing reliability requirements and mobile adhoc networking energy constraints as the determinants for the performance metrics used during evaluation. In addition to the comparison with existing approaches, we measure the performance of the distributed placement strategy under various network conditions.

Results from the comparison between our distributed placement strategy and the centralized approaches confirm our claims. The distributed placement mechanism finds near optimal placement for partial queries from a user subscription with minimal message overhead compared to the cen-

tralized approaches. For example, in some cases the distributed placement mechanism has a 48% less message overhead compared to a centralized approach used for distributed complex event processing. Additionally, the results also show an improvement in CEP reliability. Due to the intricacies of mobile computing and the limited time at hand, we did not manage to gather as much information as necessary in order to make relevant conclusions. However, early results have suggested some possible directions that can be used in future work related to this topic. Results from this investigation in general show an important impact of the partial queries semantics onto the overall distributed complex event processing performance. This and other interesting observations suggest possible directions for further work.

Contents

I Introduction and background	1
1 Introduction	3
1.1 Introduction to the problem area	5
1.2 Problem statement	8
1.3 Methodology	8
1.4 Outline	9
2 Background	11
2.1 Wireless Sensor Technology	11
2.1.1 Characteristics	12
2.1.2 Sensor data processing	13
2.1.3 Issues and challenges	15
2.2 Mobile Ad-Hoc Networking (MANET)	17
2.2.1 Routing in MANET	17
2.2.2 Power Management in MANET	19
2.2.3 Issues and challenges	20
2.3 Complex Event Processing	22
2.3.1 Event model	23
2.3.2 Query model	24
2.3.3 Distributed complex event processing in Mobile Ad hoc Networks	24
II Design and implementation	27
3 Design	29
3.1 Placement mechanism approaches for in-network CEP	29
3.1.1 Centralized placement mechanism	29
3.1.2 Distribute placement mechanism	30
3.1.3 Cluster based placement mechanism	30
3.1.4 Adaptation	31
3.1.5 Conclusion	33
3.2 System model	34
3.2.1 Data model	34
3.2.2 Mobility model	37
3.2.3 Network model	38
3.2.4 Cost model	39

3.2.5	Formal problem definition	40
3.3	Alternative one	41
3.3.1	Initial placement and event routing	41
3.3.2	Placement adaptation	43
3.4	Alternative two	43
3.4.1	Initial placement and event routing	43
3.4.2	Placement adaptation	45
3.5	Issues and challenges	45
3.5.1	Alternative one	45
3.5.2	Alternative two	46
3.5.3	Conclusion	46
3.6	Heuristic based distributed placement mechanism	48
3.6.1	The DCEP middleware	48
3.6.2	Subscription placement	50
3.6.3	Event placement	52
3.6.4	Adaptation	53
4	Implementation	57
4.1	Introduction	57
4.2	The distribute complex event processing middleware	58
4.3	Placement mechanism implementation overview	59
4.3.1	Placement mechanism meta data	60
4.3.2	Overlay message types	62
4.3.3	Initial placement for partial subscriptions	63
4.3.4	Event routing	64
4.3.5	Placement adaptation	65
4.4	Issues	66
III	Evaluation and conclusion	69
5	Evaluation	71
5.1	Introduction	71
5.2	System model	74
5.2.1	Scenario	74
5.2.2	System requirements and corresponding metrics	74
5.2.3	System entities	75
5.2.4	System entities' attributes and models	77
5.2.5	System input variables	79
5.2.6	System entities interaction and relationships	80
5.3	Simulation environment	80
5.3.1	The tools	80
5.3.2	Emulation environment setup	82
5.4	Experiment	84
5.4.1	Assumptions	84
5.4.2	System parameter values	85
5.4.3	System input variables	85
5.4.4	Simulation models	86

5.4.5	Run conditions	86
5.5	Results	87
5.5.1	Results for subscriptions with low complexity	88
5.5.2	Results for subscriptions with high complexity	92
5.5.3	Results for various network scenario	94
5.6	Conclusion	97
6	Conclusion	103
6.1	Related work and contribution of this thesis	103
6.2	Critical analysis of the results	105
6.3	Further work	106

List of Figures

1.1	D: data source, N: network node/router	6
2.1	A mica mote sensor	12
2.2	A sensor model	13
2.3	tinyDB GUI interface	14
2.4	A mobile ad-hoc network	17
2.5	Issues with energy unaware routing	20
3.1	a subscription tree	35
3.2	This image illustrates a typical node's random movement pattern.	38
3.3	MANET with a sink and three data sources for events (A,B and C)	39
3.4	A placement overlay network after initial placement	50
5.1	ns-3 main components (from www.nsnam.org)	80
5.2	ns3 components	81
5.3	simulation environment setup (obtained from: http://www.nsnam.org/wiki/)	83
5.4	The emulation perimeter and data sources location	84
5.5	Message overhead for varying mobility speeds	95
5.6	Complex event detection probability for varying mobility speeds	95
5.7	Complex event notification delay for varying mobility speeds .	96
5.8	Message overhead for varying network density	97
5.9	Complex event detection probability for varying network density	98
5.10	Complex event notification delay for varying network density	98
5.11	Major trends for the performance of the distributed placement mechanism for various network scenarios.	100

List of Tables

5.1	Network scenarios used	88
5.2	Results for centralized processing with the centralized placement mechanism	89
5.3	Results for distributed processing with the centralized placement mechanism	90
5.4	Results for high complexity subscriptions with the distributed placement mechanism	91
5.5	Results for centralized processing with the centralized placement mechanism	92
5.6	Results for distributed processing with the centralized placement mechanism	93
5.7	Results for high complexity subscriptions with the distributed placement mechanism	94

Acknowledgements

First of all i would like to begin by expressing my deep gratitude towards my supervisor Thomas Plagemann for his patient guidance and valuable constructive suggestions. I would also like to thank Phd student Piotr Kamisinsky from the Distributed Multimedia Systems (DMMS) research group for his technical support and useful critiques of this thesis. Their contribution was crucial for the successful completion of this thesis.

Part I

Introduction and background

Chapter 1

Introduction

The last two decades have been marked with advances in wireless communication technology. Moreover, as a repercussion of Moor's law, the size of computing devices has been shrinking while their sophistication grew substantially. These developments have led to advances in sensor technology and hand held devices.

Sensor technology and advances in wireless communication have enabled new kinds of applications that require real time information about the physical environment. The ability to communicate wirelessly enables sensors to be deployed in any environment providing access to information about them which is of high value for many applications from different domains.

The data produced by these sensors is typically continuous and real time. Consequently, traditional data management systems (for example, Relational Database Management Systems) are not suitable for sensor data processing. Additionally, sensor data is typically relevant for a short time and need to be consumed by sensor applications as soon as possible.

In most cases, wireless sensors are deployed in the wilderness for long periods of time. They rely on battery power and the latter determines how long they can remain operative. Unfortunately, battery power technology did not experience the same pace of development as that of computing devices. Consequently, energy constraints is one of the main challenges for sensor networks. Wireless communication has been found to be the biggest consumer of energy compared to other sensor components, thus data transmission reduction is one of the key solutions in power management or energy aware protocols at all layers of the network stack.

Data Stream Management Systems (DSMS) have been developed and successfully deployed in various application domains were the data being processed is continuous and real time. Network monitoring for traffic engineering or network security, fraudulent activities detection in financial systems are some of these application areas. The main thing these application areas have in common is the need for real time data analysis.

As mentioned earlier, sensor data is typically continuous and real time

which makes it appropriate for DSMS. However, sensor applications are typically interested in knowing when specific situations or events occurs. These event are usually at such a high level that they cannot be expressed using DSMS queries. Complex Event Processing (CEP) has been used in the last decade as the best technology for sensor data processing in order to detect higher level events of interest for most sensor applications.

The increasingly highly powered hand held devices and advances in wireless communication have also enabled significant advances in mobile computing and wide range of new applications. Some of these applications (for example, military tactical missions, disaster and rescue missions, etc..) require mobile device networks that can be formed with no networking infrastructure and without any human intervention.

Mobile Ad-Hoc Networks (MANETs) are networks of mobile computing devices which are infrastructure-less, self-creating and self maintaining. These characteristics have made MANETs popular in application areas with the requirements mentioned earlier. However, MANET technology also comes with its own share of challenges like: network nodes heterogeneity in terms of capabilities (power, transmission range, etc..), the dynamic network topology, wireless medium issues (limited availability, interference, hidden and exposed terminal issues, etc..), to name a few.

Devices used in MANETs typically run on battery power, which leads to almost the same power constraints issues identified for sensors. Thus, one of the most important techniques for efficient power consumption in MANETs is keeping data transmission minimal.

Wireless Sensor Networks (WSN) as the source of data and MANET as the communication network used to forward the data to the user applications can be used by CEP systems to enable powerful real world aware applications.

In the next section we introduce the problem we sought to investigate in more details. In Section 2 we present an outline on how we intend to implement this investigation in order to confirm the claims we make in this introduction and also gain deeper insight into this problem area. We also include a section where we present an overview of the main parts of this thesis.

1.1 Introduction to the problem area

MANETs are self creating, self maintaining and infra-structureless. Mobile nodes connected in a MANETs are typically battery powered, thus their operation duration is limited by their battery capacity. Unfortunately, advances in battery technology have yet to offer battery power which is suitable for the particular needs of mobile devices [32]. Thus, mobile nodes need to use energy efficiently in order to stay operational for longer periods of time. As a result, energy consumption optimization is central to the design and implementation of MANET communication systems[24] [4].

Experimentations have shown that wireless data transmission and reception consumes far more energy than data processing in wireless ad hoc networks. In particular, it has been shown that the energy necessary to transmit one bit of data is more or less equivalent to processing a thousand operations on a sensor device [3].

Another scarce resource in wireless ad hoc networks is bandwidth. Network nodes share the same communication medium which represents risks of network interferences and data loss.

In MANETs, nodes can move in a sudden and unpredictable manner, consequently, the network topology is dynamic and unpredictable. As a result, most of the routing protocols in MANETs consume a lot of bandwidth when processing routing information. The amount of messages transmitted during route discovery, takes up a significant part of the bandwidth that will be used for higher level data communication. Thus, the latter must minimize their message overhead in order to avoid network congestion.

Due to these issues, the reduction of wireless communication utilization can be viewed as a decisive variable in the quest to optimize energy consumption in wireless ad hoc networks [4].

Sensors are used to detect, sense or measure physical stimuli from the real-world environment. However, application domains like military tactical support or Emergency rescue missions are interested in complex events which emanate from the correlation or filtering of sensor data.

Data stream management systems are used to aggregate, correlate and filter sensor data samples in order to detect complex events from them. However, CEP technology is better suited for some of the application domains due to its expressiveness. CEP consist in using predetermined rules or queries in order to detect complex events in a near real time manner.

Together with CEP, sensor networks represent a powerful means to detect events of interest in many application domains [17].

The main idea about CEP in sensor network is that a user typically express her event of interest in the CEP engine's query language. The CEP engine uses the user's queries in order to filter or correlate sensor data.

Ultimately, the main purpose of the CEP engine is to perform correlations of the sensor data from the sensors in order to detect complex event matching the user interest expressed through the submitted queries. In its

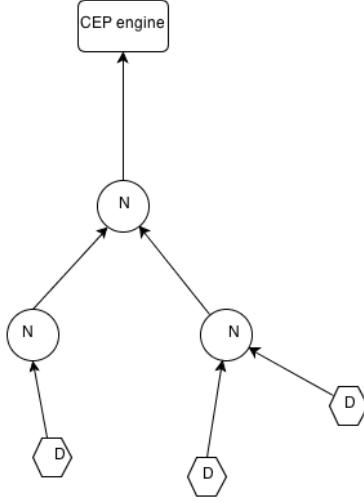


Figure 1.1: D: data source, N: network node/router

simplest form, the CEP system is centralized see Figure 1.1.

All the sensor data is sent to the CEP located at the central node, also called sink, for processing. The sensor data is delivered to the sink in a hop by hop manner typical for ad hoc wireless networks. This means that nodes inside the network must collaborate in order to deliver the events to the sink. This approach is inefficient and wastes network resources for the following reasons:

1. Sensors typically produce a continuous stream of data and only a small portion might be of interest for the user. Furthermore, part of sensor data processing consist in merging data from different sources and the output is typically less than the input. Consequently, scarce network resources will be used to transport and process irrelevant data.
2. The continuous nature of sensor data and the fact that all data from sensors is converging towards one node can quickly saturate the network's bandwidth leading to network congestions.

Due to the issues related to the centralized CEP scheme, and the fact that sensor data sources are typically spread throughout the network, in-network processing technology has been proposed as a resource efficient solution for sensor data processing in general. In an in-network CEP scheme, the queries submitted by the user must be divided into smaller queries or partial queries which must be processed in order to produce complex events that match the original user query. The partial queries that are constituents of the original user queries are distributed among network nodes running CEP engines for processing. The CEP engines must collaborate in order to process the partial queries appropriately and be able to detect complex events of interest for the user. The distributed processing of the query should yield the same result as if the query submitted by the user were processed by a single CEP engine.

The main task of a CEP engine is event correlation in order to detect underlying complex event patterns. Thus, the amount of events output is usually not the sum of the input events. For this reason, the location of a CEP processing a specific partial query in the network effects not only the amount of events transmitted, but also the hop count the events have to travel. In this thesis, the amount of event transmitted and the number of hop count those events have to travel is an important part of overall cost of processing a user query. Thus, the mapping of the partial queries on various CEP engines in the network has a high impact on the overall cost of answering the original user query [12] [23]. This makes the mechanism for partial queries placement a central function for reducing data transmission and enable an energy efficient CEP in ad hoc sensor networks.

A placement mechanism for distributed CEP (DCEP) seeks to find the optimal placement for each partial query in order to minimize the cost of processing a subscription.

The process of finding the optimal placement for a partial query and sending it to the appropriate node for processing, introduces additional computational and transmission costs. This is of course to be considered when evaluating the overall cost of processing a user query with a specific placement mechanism. This is important since placement mechanisms which consume more network resources than what they save through the optimal placement of partial query should be avoided. Thus, one needs to include this cost when evaluating the overall cost of processing a user's query in order to find the true and accurate incentives related to using a certain placement mechanism.

The placement mechanism can be centralized or decentralized. In a centralized scheme, a central node, usually the node that receives a query from the user, can perform the placement of all derived partial query inside the network. On the other hand, in a decentralized scheme, placement of the partial query is performed in a distributed manner throughout the network. Each of these approaches has its advantages and disadvantages.

Due to the nodes' mobility and changes in the input data rate overtime, the initial placement performance will eventually deteriorate [35, 12]. Thus, a placement mechanism should be dynamic in such a way that it can re-evaluate previous placement decisions and determine whether to adapt any of them to suit current network and data traffic conditions. However, the need for placement adaptation must always be balanced with the inherent message cost unless it lowers network performance instead of increasing it. For example, it wouldn't be necessary to update the entire placement plan when only parts of it are affected by the changes in the network [12].

Again, the need for a dynamic placement mechanism introduces additional computational and transmission costs to the overall cost of processing a user's query. There should always be a balance between finding the optimal replacement for a processing node and the message

overhead impact on the overall processing cost of the user's query. Ultimately, the main purpose is to minimize message transmission and save energy in the network.

1.2 Problem statement

This thesis investigates different placement strategies for DCEP in MANETs. A placement strategy or mechanism should be able to perform the following tasks:

- Find an optimal placement for each partial query in order to minimize data transmission in the network and save energy.
- Perform event routing between network nodes processing related partial queries and successfully deliver the complex events to the sink.
- Be able to adapt the initial partial query execution plan to the dynamic topology at minimal data transmission cost.

We argue that a distributed placement mechanism can reduce the amount of messages transmitted during complex event processing and thus reducing energy consumption. This claim is based on the assumption that a central node cannot have all the topology information necessary to produce an optimal partial query execution plan. We further claim that this can be achieved with no negative impact on CEP reliability.

To confirm our claims, we design, implement and evaluate a distributed placement mechanism. The evaluation of the placement mechanism comprise two parts. The first part compares the performance of the distributed placement mechanism with that of centralized approaches. The second part evaluates the distributed placement mechanism for various network scenarios.

1.3 Methodology

First we use existing literature about CEP, data processing in wireless sensor networks and MANET in order to identify the main issues, challenges and requirements that are related to CEP in MANET. This should help us develop our own distributed placement mechanism with the identified challenges into perspective.

The process of developing the distributed placement will provide us with valuable hands on experience with the area of inquiry. This will extend the knowledge gained from the literature.

We then evaluate the distributed placement mechanism together with existing centralized approaches from [17]. This evaluation is based on pre-defined performance metrics in terms of CEP reliability and identified requirements for data processing in MANETs. This should help us support

our claim about the incentives of distributed placement strategies over centralized approaches. We also evaluate the performance of the placement mechanism in various network scenarios in order to gain further insight into placement strategies for DCEP and set direction for further investigations in this area.

1.4 Outline

Background In this chapter we introduce the main topic areas that constitute the foundation for the work done in this thesis. The motivation, characteristics, issues and requirements for the main topic areas are identified.

Design and implementation Using existing literature on the problem area and the identified characteristics, issues and requirements from the previous part, we design and implement a distributed placement mechanism.

Evaluation and results In this part, the distributed placement mechanism is evaluated by comparing its performance to that of centralized approach. Additionally, we evaluate the performance of the distributed placement mechanism in various network scenarios. Finally, we conclude this part with a discussion about the results and how they relate to the goal of this thesis.

Conclusion In this part, we discuss related work and highlight the contribution of this thesis. We also present a critical analysis of the results and the thesis in general. Finally, we propose interesting directions for further work related to what was done in this thesis.

Chapter 2

Background

In this chapter we present the wireless sensor technology along with its characteristics and challenges. This should provide some insight into the main sensor data processing requirements. Afterwards we address the topic of Mobile Ad hoc Networking. Here we focus on the main characteristics related to routing and power consumption as they are closely related to the issues addressed by this thesis. We also identify the main challenges and issues that will serve as a guide line for later sections. Based on the issues and challenges identified in both wireless sensor technology and Mobile Ad hoc Networking sections, we introduce the CEP paradigm and DCEP. In this section, we address the main characteristics of CEP from which we derive the importance and need for an efficient placement mechanism for DCEP in MANETs.

2.1 Wireless Sensor Technology

A sensor is an electronic device that detects, senses or measures physical stimuli from the real-world environment and converts it into analogue or digital form [11] . These stimuli represent events or states that can be of interest for various real-world aware application domains. Some of the application domains are:

- Health care: Heart rate monitoring
- Environmental monitoring: temperature, light
- Emergency and rescue missions
- Military tactical missions
- Location sensing
- Video surveillance etc...

As an example, sensors can be deployed in a wildfire disaster area in order to monitor their surroundings' environment temperature. This information is crucial for the fire fighters to plan and coordinate their operations.



Figure 2.1: A mica mote sensor

2.1.1 Characteristics

A wireless sensor device can have the following components:

- A processing unit which manages the other components and performs necessary computational tasks.
- A radio-communication/transceiver unit which connects the wireless sensor device to the network by sending and receiving data.
- A memory unit both for short term (RAM) and long term storage (EEPROM, ROM, etc.).
- A sensing unit which performs the task of sensing physical stimuli. Usually, the sensing unit is made of two parts: one or more sensors that perform the actual sensing and an analogue to digital converter which transforms the sensed stimuli into digital data that can be processed by the processing unit.
- Actuator which can be used to manage the power and sensor units.
- Power unit which provides power to the wireless sensor device.

Figure 2.1 shows a mica mote sensor.

The processing unit (micro-controller) coordinates the sampling of the sensing unit(s) and sends packets of data to the transceiver unit which can send it to other network devices. Various controller architectures can be used for the processing unit. For example, Micro-controllers, Micro-processors Field-Programmable Gate Arrays(FPGAs) or Application Specific Integrated Circuits(ASIC). Each of these controller architectures comes with its own advantages and disadvantages in terms of flexibility, performance, energy consumption and costs [18]. As an example, micro-controllers are preferred in wireless sensor networks for their ability to go into sleep mode (only parts of the controller are active) which helps save energy. They can also be easily connected to various types of sensors. Some existing micro-controllers are: Intel StrongARM, Texas Instruments MSP 430, etc.

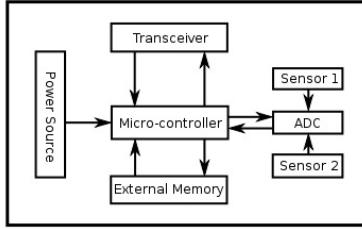


Figure 2.2: A sensor model

Programs used on these wireless sensor devices are typically stored on a flash memory or a ROM. In addition to the non volatile memory, wireless sensor devices also have volatile memory (SRAM) which is used to store variable data, sensor readings, packets from remote nodes, etc.

The radio communication unit is used to send a receive data to and from other nodes in the network. It is usually made of one device (the transceiver), but can also be made of two devices: a transmitter and a receiver. Various transmission medium can be used: radio frequencies (typically between 433MHz to 2.4GHz), optical communication, magnetic inductance and ultrasound. Radio frequency is usually preferred transmission medium in most cases.

Different types of battery are used inside wireless sensor devices. The kind of battery used determines the overall performance of the wireless sensor devices. Lithium batteries are preferred as they tend to have a longer shelf-life.

Some of the existing sensor devices are: the "Mica Mote" family (see Figure 2.1), EYES (Energy Efficient Sensor Networks) devices, BTnodes, Scatterweb, etc.

2.1.2 Sensor data processing

Sensors are used to enable applications that are real-world aware without human intervention [16]. To achieve this, sensors devices are spread throughout the area of interest where they can monitor their environment. Typically, sensors form an ad hoc network with one or more gateway nodes also called sink(s)

One typical operation in sensor networks is interest dissemination [1]. A user sends her interest to the sink, or the whole network and expects to be notified if events that match her interests are detected by the sensors. The user needs an application with an interface where she can express her interest in a declarative way without the necessity to know the location of the sensors. Additionally, the application should also enable the user to receive notifications when her interests are met. In this sense, the sensor network can be viewed as a (dynamic) database. Indeed, the act of expressing one's interests in a specific outcome in the physical environment is similar to for-

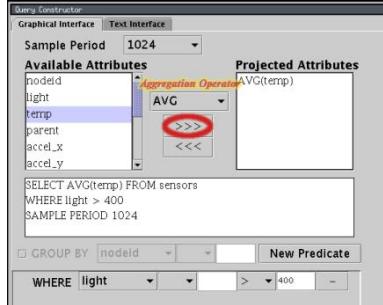


Figure 2.3: tinyDB GUI interface

mulating queries for a database [18]. Consequently, one can regard the sensors as a virtual table to which relational operators can be applied. As an example, tinyDB is a query processing system developed for sensor networks data processing. As it appears in Figure 2.3, tinyDB offers a simple GUI that can be used to formulate a user's interest in a SQL like query language.

However, sensor networks require different approaches to data processing different from traditional database management systems. This is due to the continuous, real time and unpredictable nature of sensor data.

Data Stream Management Systems

Sensors deployed for monitoring purposes usually produce a large amount of continuous data. In many implementations, from network traffic monitoring to security monitoring, sensor data need to be processed in a timely manner. In other words, this data is only relevant for a short time and should thus be processed immediately without any "transition storage". The availability of large amount of data to be processed without the possibility to store it is not suitable for traditional Database systems. These systems rely on the fact the data they are processing is stored on disk and are thus tuned and optimized for this situation. Furthermore, the processing of data in traditional Database management systems is triggered by a human submitting a query to the system. In contrast, sensor data processing must be driven by data availability which makes traditional Database Management Systems unsuitable for this kind of processing model. Furthermore, traditional Database Management Systems process persistent data while sensor data is continuous and must be processed in a timely manner.

There has been an attempt to extend Database Management Systems in order to make them able to trigger processing based on predefined events; Active Database Systems [26]. However, these systems cannot keep up when the rate of incoming events increases. Data Stream Management Systems have been developed to deal with the limitations of Database Management Systems and Active Database Systems by enabling processing of continuous streams of data. DSMS borrow many features

from traditional Database Management Systems.

Data models The data are real time and continuous arriving in some order, possibly from different sources. The stream of data is composed of items/tuples with specific attributes and values. The stream of data contains useful but also useless information which needs to be filtered out. Only selected parts of data in these streams might be stored, otherwise, nothing is stored. This data is low level and usually needs to be aggregated in order to produce relevant information that can be used. As we shall see, various techniques are used to deal with the continuous and unpredictable nature of this data while being able to produce useful information. However, despite powerful techniques used to make sense of these streams of data, the data model for DSMS is limited in areas where there is a need for data with some kind of semantics. The event model is a major step forward as it allows complex description of events which opens doors for more powerful query languages.

Queries The data model discussed above requires a rethinking of the way data is usually processed by Database Systems. For this reason, as opposed to traditional database systems, DSMS queries are continuous. Predefined queries are continuously applied to incoming streams of data. Moreover, while traditional database uses ad hoc queries, data stream queries are stored. Blocking operators are also used in Data Stream Management Systems but are hard to deal with since normally only one pass over the stream is possible (data is not persistent). Techniques like windowing, batch processing and others, are used in order to deal with blocking operations. Moreover, we need to reduce the data in order for it to fit in memory and be processed through the use of techniques like Summary structures. Despite the use of powerful mechanisms to process the data, DSMS queries fall short in providing high level information from processed data.

2.1.3 Issues and challenges

One of the most important challenges with wireless sensors devices is power consumption management[2]. The performance of sensors relies heavily on their battery capacity. This is due to the fact that wireless sensors devices are usually deployed in remote areas where they must be able to keep functioning for long periods of time without being recharged. Because the communication unit has been found to be by far the biggest consumer of energy [2], Systems developed for sensor data processing must be able to minimize data transmission even if it might lead to more CPU activity.

Sensors usually use the Industrial, Scientific and Medical (ISM) band for data transmission. The ISM band is preferred for its huge spectrum allocation and global availability, and it is free [1]. However, sensor data transmission suffers from interference from other devices (probably with

more powerful transmission devices) using the same frequency bands due to the fact that the ISM frequency bands are unregulated. This issue comes as an additional challenge to the usual intricacies related to wireless communication [2].

While DSMS represent a considerable advance in building suitable data management systems for continuous streams of data, they still have some limitations. In essence, results from queries applied to the streams of data are not expressive enough. The data model used in DSMS does not allow the user to formulate queries that are powerful enough to detect high level events that might be a composition of related events from different sources. The selection of events can only be based on attributes and values of the data items. Moreover, these values are limited to low level semantics like timestamp, temperature readings etc.. (eg. tinyDB queries). Additionally, DSMS systems cannot detect complex event patterns involving sequences and ordering relations [13].

In traditional database systems, an execution plan is always produced from the user's query. Execution plan optimization schemes are even used in order to enable an efficient query processing. Similarly in sensor networks, an execution plan must be produced from the user's query or any other language used to express her interests. However, due to the fact that the sensors producing the data are typically scattered over wide areas, the execution plan might have to be distributed over the network. As we will see in later sections, the task of assigning partial queries to network nodes for processing is similar to the task assignment problem which has been found to be NP-complete. Additionally, unlike in traditional database systems, the execution plan must take into consideration additional variables like: communication cost, power consumption, mobility, etc.

Finally, the actual physical environment where sensor devices are deployed can also impedes the sensors' operations.

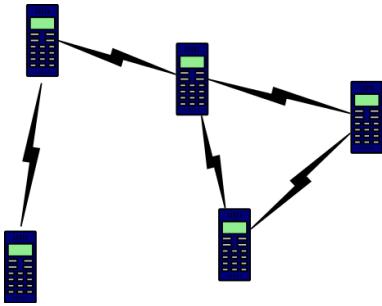


Figure 2.4: A mobile ad-hoc network

2.2 Mobile Ad-Hoc Networking (MANET)

The miniaturization of computing devices, the advances in wireless communication has led to a wide spread availability of low cost wireless devices with high computational power. This has led to the popularity of mobile computing and new application areas. Some of these application areas are:

- Emergency rescue missions,
- military tactical missions,
- sensor networks, etc..

Most of these applications areas require the ability to create communication network in the absence of network infrastructure. This is usually due to the fact that it is impractical, expensive or impossible to set up networking infrastructures [32].

Additionally, applications like Emergency and rescue missions or military tactical missions require spontaneous and fast network creation without human intervention. Moreover, they also require the ability to stay connected and interoperating despite human mobility.

A MANET represents a system of wireless mobile nodes that can freely and dynamically self-organize into arbitrary and temporary network topologies, allowing people and devices to seamlessly internetwork in areas without any pre-existing communication infrastructure [22].

2.2.1 Routing in MANET

Due to the lack of infrastructure and the limited range of wireless communication, nodes in MANETs perform a multi hop communication between them. This means that MANET nodes act both as end systems and routing devices.

The routing architecture is typically flat or hierarchical. Most routing protocol in MANET use the flat routing architecture where all nodes participate in routing and are equal. All nodes know about each other and

store information about the entire network topology (more on this later). There is a storage and communication overhead inherent to the flat model which impedes the system's scalability.

Hierarchical routing protocols use techniques like clustering in order to increase scalability(more on this later..).

Traditional routing protocols based on link state and distance vector distance cannot be applied to MANET due to the nodes mobility, constrained resources, network partitioning etc. Thus different routing protocols have been developed for MANET. These protocols are usually classified in three groups:

1. Proactive routing protocols
2. On demand or Reactive routing protocols
3. Hybrid routing protocols

This classification is based on the mechanisms used by the routing protocols to gather and maintain routing information on mobile nodes. An other classification of routing protocols is based on the nodes' role in routing. Two main groups emanate from this classification:

1. Uniform routing protocols where all nodes have equal responsibilities in the network.
2. Non-uniform routing protocols where some nodes are selected to perform routing function over the entire network. This is done in hierarchically structured networks and the main purpose here is to deal with scalability issues in MANET. Non-uniform routing protocols are further divided into three groups:
 - (a) Zone-based hierarchical routing where nodes are organized into different zones with selected nodes to forward data between zones.
 - (b) Cluster-based hierarchical routing: Special nodes called cluster heads are periodically elected and each is responsible of a subset of nodes in the network. Only cluster heads know about each other and data is forwarded between them through cluster gateways.
 - (c) core-node based routing: The core nodes form a backbone in the MANET and perform special functions, such as routing path construction d control data packet propagation.

On demand routing protocols On demand routing protocols only calculate destination path when the path is requested by a local application. This has the advantage of limiting network and processing overhead while increasing the time it takes to find the path to a destination. However, if requested paths are saved and the node mobility is not to high, the delay only happen for first time destination path requests.

Proactive routing protocols Proactive routing protocol calculate routing information constantly without waiting for any request from local applications. At any time, every node has access to the information about the entire network topology. However, this comes at a price. The network is constantly flooded with control information when nodes' mobility is high. furthermore, the higher the network size gets the more information must be exchanged. Obviously, this category of routing protocols has its own advantages and disadvantages and it all depends on what one wants to achieve. For example, routing protocols like OLSR use special message flooding techniques intended to significantly reduce the amount of control messages during routing. This makes OLSR attractive for applications that require low message delivery delay like complex event processing.

Hybrid routing protocols Hybrid routing algorithms tend to combine the other two types of routing protocols by periodically acting in a proactive way and otherwise calculating routes on demand. This is done in order to try bring together the advantages of both proactive and reactive routing schemes.

2.2.2 Power Management in MANET

Since nodes that operate in MANET are battery driven, power conservation is one of the central issues in such networks [22].

A node's battery power is typically shared among various hardware components like: display monitor, wireless networking interface, the central processing unit, memory unit, etc. However, the wireless networking interface card has been found to consume 10-50% of overall system energy [22]. Additionally, data transmission has been found to consume more energy than data reception. As a result, the wireless networking interface usually supports different operation modes (sleep, receive and transmit modes) in order to minimize power consumption. Higher level services and applications should cooperate with the wireless networking interface card in order to determine when to tune between different modes when appropriate as a mean to save energy. MANET software should also reduce unnecessary transmissions as much as possible [22].

Generally, power-conservative protocols are divided into two categories[22]:

- Transmitter power control mechanisms and
- power management algorithms

Transmitter power control refers to techniques used to tune wireless transmission powers to the proper range. Since power consumption increases with the transmission range, power control can be used to save energy. Additionally, reducing the transmission range can reduce radio interference increasing the bandwidth available for network traffic. However,

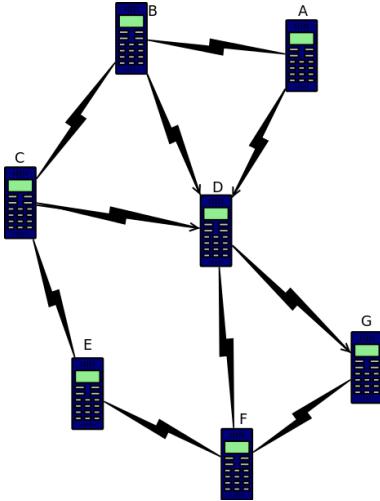


Figure 2.5: Issues with energy unaware routing

short transmission ranges can introduce additional issues like network partitioning [22].

The category for power management algorithms includes: MAC layer power management, network layer power management and application layer power management.

The MAC layer power management is considered crucial for the overall device's power consumption. For this reason, a lot of research has been conducted in order to develop efficient MAC layer power management algorithms. For example one proposed approach from [8] is to estimate the probability that a particular frame will be transmitted successfully and only send it if this probability is "high enough".

At the network layer, routing protocols also need to be power aware unless they drain the network's power resources. Routing protocols in MANET should be based on shortest cost not just shortest hop [22]. In other words, the shortest cost calculation should be energy aware. As an example, consider Figure 2.5, if nodes A, B, and C only consider the route with the least number of hops, node D's power will be quickly drained. Instead, the routing protocol should be aware of this kind of situations and avoid them when possible.

At the transport layer, TCP is ill suited for the volatile MANET environment. Because it was not developed with MANET characteristics in mind, the TCP protocol leads to poor performance and high energy wasted through unnecessary retransmissions.

Higher level applications and protocol can and should also be power aware by minimizing their message cost as much as possible.

2.2.3 Issues and challenges

The communication performance of a network is crucial for the higher level systems reliability. This means that processes running on different nodes

in the network should be able to exchange messages at high speeds especially for real time data processing. This means that routes to remote nodes need to be available when needed and up to date. Additionally, intermediate nodes between the source and destination of a message must be able to quickly route the message towards its destination.

These requirements are difficult to reach due to the dynamic network topology of MANET and the volatile nature of the nodes and the wireless communication medium.

The dynamic nature of MANET topology makes it difficult for routing protocols to keep updated and consistent route information while nodes are moving in unpredictable and sudden manner. This means that route information must be constantly discarded and new routes must be found for higher level applications and protocols. The choice between proactive and on demand routing protocols in MANET is not an easy one since they both have their advantages and disadvantages. The proactive approach comes with a high message cost while the on demand approach can introduce longer communication delays. Usually, the choice is made based on higher level protocols performance requirements.

MANET suffers from the limited resources of the network nodes (especially energy) and the intricacies of wireless communication. This represents additional challenges not only to the routing services but to the higher layers protocols as well. For example, network partitions caused by node failure, wireless interference or mobility can lead to system availability issues (inaccessible network services for example).

The operational life time of the mobile nodes and indeed the entire network depends on how well power consumption is managed in the network. As mentioned earlier, various power management schemes can be implemented on all layers of the networking stack. Consequently, new MANET protocols and application must be developed in such a way that they minimize power consumption. However, the need to reduce power consumption can easily crash with MANET systems reliability requirements. For example, power aware routing might provide longer routes that might increase data transmission delays. Using on demand routing approach can also lead to high message delivery delay. At the transport layer or in middleware the need to increase periods between data retransmissions in order to save energy might lead to higher message delivery delays. The task to choose long enough but not too long periods between retransmissions can be very difficult considering the unpredictable and sudden movement of the nodes and the unstable nature of the wireless medium.

Other techniques like replication used for system reliability can be conflicting with the need to minimize data transmission in order to optimize power consumption.

2.3 Complex Event Processing

As mentioned earlier, users of sensor networks applications need an interface where they can express their interests in a declarative way without the need to know about the location of the individual data sources. Therefore, declarative programming is the preferred approach in sensor data processing.

Additionally, the users' interest are typically specific events in the physical environment and they would like to be notified when these events happen. Consequently, the data they are interested in is not yet available, therefore sensor data must be applied to the users' queries not the other way around which is typical in traditional databases systems.

Moreover, the continuous and unpredictable nature of sensor data makes it practically impossible to store it before processing considering the limited resource availability in sensor networks. Furthermore, users of sensor networks applications typically want to be notified about events right after they happen (in real time), which means that sensor data must be processed in a timely manner.

We have seen already that DSMS have been successfully used to process sensor data. However, they are limited in terms of what kind of events the system can detect. More specifically, DSMS systems cannot detect complex event patterns involving sequences and ordering relations [13].

In publish-subscribe systems allow users (subscribers) to express their interests into a more expressive rule language (subscriptions). The data items produced by the data sources or observers of events(publishers) is applied to the subscriptions and users are notified when the events the subscribed for are detected. However, publish-subscribe systems are still limited by the fact that they only process one event at a time missing out possible relationships between events from different sources. Events in Publish-Subscribe systems can be filtered based on:

- channel: Events are published on different channels and subscribers subscribe to those channels. Notice that the actual filtering is based on channels, possibly sources, rather events.
- Topic: this filtering model allows more expressiveness as one can for example describe events on different level of hierarchy.
- Content: this model adds to the expressiveness of topic based filtering by allowing further filtering of topics based on their content.
- Type: this model is similar to content based filtering but allows better integration with programming languages.

As it appears, publish subscribe systems do not allow event composition where events are described from other events occurrences, ordering or patterns. CEP operates not only on sets of events but also relationships between events [25]. CEP Systems add an extension to Publish-Subscribe model by allowing subscribers to express their interest in composite events

[26].

Complex event processing (CEP) has evolved into the main paradigm for various applications from areas like financial and battlefield applications. It is the paradigm of choice for monitoring and reactive applications [9]. This includes but not limited to sensor networks applications.

CEP decouples the information sources and the information consumers, enabling the declarative programming required for sensor networks applications. More specifically, the information consumers do not need to know about the location of the information sources and can thus express their interests in a declarative manner. Similarly, the information producers do not need to know anything about the location of the information consumers.

Additionally, through aggregation and composition of events from different sources, CEP offers a powerful means to detect high level and complex events. This suits well the need for a more expressive rule language that sensor networks application users can use to express more interests in more abstract events that offer a deeper insight into the situations of interest. For example, in a home environment scenario, a CEP engine like Esper can detect high level events like the fact that a person is cooking.

Before delving into complex event processing in sensor network and MANET in particular, we first explore the event and query model for CEP in order to gain further insight into the characteristics of CEP.

2.3.1 Event model

The word event is used in various instances of everyday life. Thus, it can have different meaning to those using it. According to the online Oxford dictionary, an event is; a thing that happens or takes place, especially one of importance. An event can also represent a particular type of action or change that is of interest to a system, occurring either internally within the system or externally in the environment with which the system interacts [10].

In the case of sensors, the actions or change of interest are rather external to the system. Furthermore, if we consider the set of all states or stimuli that the sensor is supposed to measure or sense in its physical environment, an event would then be any member of the sub-set of states whose values/characteristics correspond to a predefined threshold, margins or even patterns. Indeed, the predefined thresholds, margins and patterns represent the sub-set of things happening, that are of importance for us.

In the attempt to describe or model an event, it can be helpful to classify events either as atomic or complex. An atomic event is an event that cannot be divided into any other event [16]. In essence, an atomic event is an indivisible member of the set of events that are sensed by the sensor. On the other hand, a complex event can be seen as a composition of two or more

events from same or separate source(s). In other words, a complex event can be seen as a set of atomic events that are consecutively or simultaneously related [16].

A more general and formal way of describing an event is achieved by assigning properties to events [33]. Event properties can be:

- Temporal: This corresponds to the physical or logical timestamp of the event.
- Spatial: Spatial properties of an event correspond to its source for example.
- Informational: Informational properties of an event provide specific information related to that particular event.
- Experiential: experiential properties of an event represent its relationship with earlier events and or event from other sources.
- Structural: Structural properties of an event are used to determine the event's level of abstraction or maybe its position in the tree hierarchy as discussed earlier.
- Causal: Causal properties of an event describe or determine the event's causal relationship with other events.

2.3.2 Query model

The query model in CEP is similar to that of DSMS in that they both are inspired by declarative languages. This means that the user or programmer focuses only on what she wants not how she will get it. In other words, queries that are applied to the streams of events describe the event patterns of interests not how to get those patterns.

Furthermore, many of the mechanisms used in DSMS are reused in CEP. However, due to a different data model, CEP adds new capabilities to their query model in order to easily describe and filter complex events. Queries must be able to filter events not only based on their informational properties, based on event patterns as well. These patterns relate events to each other through their temporal, spatial, experiential, structural, causal and even informational properties. Clearly, the mechanisms mentioned earlier for DSMS are not enough to achieve this level of expressiveness. Streams of events pass through predefined event queries which use their powerful language construct to filter complex events.

2.3.3 Distributed complex event processing in Mobile Ad hoc Networks

In some sensor networks applications like Emergency and rescue missions, the sensors deployed in the environment to monitor need to send data

about the sensed physical stimuli to the sink. However, typical to these situations is the lack of network infrastructure. Therefore, the MANET formed by wireless devices held by the rescue personnel is usually used to forward data from sensors to the application node.

As mentioned earlier, resources are usually scarce on MANET wireless devices. More specifically, we have seen that wireless devices in MANET have limited energy resources. Furthermore, data transmission has been found to consume far more energy than the other hardware components in the wireless devices. Therefore, it is necessary to limit data transmission as much as possible. For this reason, in addition to being reliable, complex event processing must also be energy aware by minimizing message transmission.

As mentioned earlier, sensors are usually scattered all over the area that must be monitored. Additionally, sensors typically produce a high volume of fine grained data. Consequently, a centralized complex event processing scheme with a CEP engine at the application node would be inefficient in terms of energy consumption. the high amount of sensor data would quickly drain the network's energy resources. Moreover, considering the fact that sensor data is typically aggregated and filtered, a portion of it is discarded by the CEP engine. Thus, the need to process sensor data earlier and reduce the data that is actually forwarded through the network.

The stepwise correlation of events can help reduce the message load while enabling CEP scalability. This can be achieve by distributing the subscription processing over several nodes in the network. Essentially, a subscription is split into more than one smaller parts which can be assigned to nodes in the network and processed independently.

The task of assigning a group of related partial subscriptions to nodes in the network is similar to the task assignment problem which has been found to be NP-complete [6] [34].

Additionally, determining which node should process which subscription determines the overall cost of processing a user's subscription [12] [23]. This cost includes the message cost related to placing the subscription's parts inside the network in addition to the message cost for event forwarding. This makes the placement mechanism central in the quest to minimize energy consumption in addition to CEP reliability.

Part II

Design and implementation

Chapter 3

Design

In this section we design a distributed placement mechanism that will be used to assert our claims for this thesis and further investigate placement strategies performance for DCEP. The next section present a discussion about possible approaches for placement strategies in CEP. Section 2 present the system model which represent the foundation for the design and implementation of the distributed placement mechanism. Section 3 and 4 will explore two distributed approaches for placement mechanism. Section 5 discusses the issues and challenges observed from the two distributed schemes for placement. Section 6 will outline the chosen placement scheme and present its detailed design features.

3.1 Placement mechanism approaches for in-network CEP

In this section we briefly discuss different approaches for placement mechanisms.

3.1.1 Centralized placement mechanism

In a centralized placement mechanism scheme, a central node (usually the node which receives the query from the user) uses network topology information to find the optimal placement for each of the partial subscription derived from the user's query.

The placement mechanism is straightforward and easy since it is based on a single network topology snapshot despite the underlying dynamic environment. Furthermore, there is no message overhead related to finding the optimal placement for the partial subscriptions.

The centralized approach is not scalable since the node performing placement needs to know about the entire network topology in order to find the optimal placement for partial subscriptions [24].

3.1.2 Distribute placement mechanism

In a distribute placement mechanism scheme, network nodes collaborate in order to find the optimal placement for all the partial subscriptions from a user subscription. Consequently, the distributed approach is able to find the optimal placement plan.

The problem with this approach is that it requires additional message overhead related to finding the optimal placement for partial subscriptions [24]. Thus, the inherent data transmission risks discarding the incentives of performing a distributed placement in order for find a more optimal placement for partial subscriptions.

In cases where synchronization between nodes is required in order to find the optimal placement for a subscription [34], the dynamic topology environment for MANET might make it almost impossible to perform placement [23].

In some distributed implementations ([34]), all the network nodes participate in the placement process while only part of them might be eligible as partial subscription processor, considering the location of the events data sources . This could be rather unfortunate since those nodes that are not eligible for event data processing could be temporally switched off in order to save energy. One of the techniques used to save network nodes power consist in turning some of them off alternatively while making sure the network is not partitioned and data processing performance is kept in balance with the aimed level of energy consumption [32].

Some MANET routing protocols use network clustering as a solution for typical network flooding used to build routing tables information. This techniques is also exploited by some MANET energy management schemes that use cluster heads to switch on and off their slave nodes alternatively and thus saving energy.

3.1.3 Cluster based placement mechanism

Clustering technique consist in creating a virtual partitioning of a mobile ad hoc network. This can be done based on nodes connectivity, nodes' mobility, etc. The goal is to form an overlay of selected nodes called cluster heads which are connected to each other throughout the network. The rest of the network nodes can only communicate within their respective virtual clusters with the cluster head acting as a coordinator.

MANET clustering enables high scalability in MANET data processing. The placement algorithm could now involve only the cluster heads allowing DCEP in large scale sensor networks. Moreover, since in some clustering scheme, the cluster head is chosen based on its degree of network connec-

tivity, one could consider performing a centralized placement scheme inside the virtual clusters. A centralized scheme would yield much less message overhead related to finding optimal placements for partial subscriptions.

The hierarchical network topology which results from network clustering makes it possible to minimize the number of nodes that are needed in order to find optimal placement for partial subscriptions.

However, the main draw back about network clustering is its inherent message overhead related to cluster maintenance.

3.1.4 Adaptation

As mentioned earlier, a placement mechanism should be able to adapt its execution plan over time due to the inevitable changes that occur both in the network topology and data traffic patterns.

Based on the criteria used to perform the initial placement of partial subscriptions, the placement mechanism should be able to constantly check whether the execution plan is still optimal.

The adaptation scheme can be performed in a centralized or decentralized manner. Furthermore, the adaptation scheme is not limited to the criteria used during initial placement when evaluating the optimality of the execution plan. However, in this project we stick to the criteria used during initial placement. Additionally, it is crucial for the adaptation scheme to balance between keeping an optimal or near optimal execution plan at all time and keeping low the message overhead related to placement adaptation.

In a centralized adaptation scheme, one node maybe the application node could be responsible of performing placement adaptation based on information gathered locally or from nodes processing partial subscriptions. An adaptation scheme based solely on information from one node requires that the latter is the one that performed the initial placement of all partial subscriptions in the first place. Thus, this scheme would be part of a centralized placement mechanism. Consequently, it the scheme would suffer lack of scalability and poor placement decisions.

However, if the adaptation scheme uses information gathered from all nodes processing partial subscriptions, it can be part of a centralized or decentralized placement scheme. Furthermore, such an adaptation scheme would make decisions based on more accurate data. Every time a node processing a partial subscription detect change in predefined metrics (data rate, topology, etc.) it would send a notification to the application node. The latter would then decide what to do based on a predefined algorithm. It is possible in this scheme to make optimal placement decisions due to the fact that the decisions are made based on the overall execution plan not just

the partial subscription affected by the current change. If updating the partial subscription placement will not be beneficial to the entire execution plan, the latter is left as it is. Otherwise, the partial subscription placement is updated as well as additional partial subscriptions that might be affected. This scheme is also simpler and maybe better suited in a MANET environment since the entire adaptation mechanism is done by one node thus avoiding the complication of more than two nodes communicating to update a partial subscription placement. More specifically, the adaptation needs to be performed quickly in order to avoid situations where there would be more than one execution plans at one point in time. For example, this would be the result of more than one adaptation taking place at the same time.

A problem with a centralized scheme is that it would have a high message overhead due to the nodes constantly sending change notification messages to the application node. Furthermore, the centralized approach is not scalable.

Moreover, the execution plan made by this adaptation scheme will not be optimal. Nevertheless, the centralized scheme has the advantage of being able to enable adaptation mechanism avoid intricacies related to inconsistent execution plans as viewed by processing nodes.

A decentralized scheme can be part of a centralized or decentralized placement mechanism.

One approach to perform placement adaptation with a decentralized scheme is to let each node monitor changes that affect each of the partial subscriptions that are placed locally. This way, whenever, change is detected, the node re-asses placement for the affected partial subscription. If it is no longer suitable to process the partial subscription it initializes placement of the latter in a centralized or distributed way.

If the placement is done in a centralized manner, the node processing the affected partial subscription simply determines which other node is more suitable to process the partial subscription. When found the partial subscription is sent to the new node and the old processor or the new processor can update the other nodes concerned by the change. This scheme's advantage is that it performs adaptation quickly and thus avoid issues related to inconsistent execution plans. Furthermore, because other nodes impacted by the placement adaptation are notified, the resulting execution scheme will still be optimal. However, the cost of a partial subscription adaptation as a result of change in the network cannot be predicted by the node that initialize the adaptation process. This is due to the ripple effect related to the partial subscription adaptation.

If the placement is done in a decentralized manner, the node processing the partial subscription affected by change initializes a distributed placement for it. While this approach could find the optimal placement for the affected partial subscription, it might take some time due to the mobile topology. Consequently, different adaptations routines might overlap each other and cause inconsistencies in the execution plan.

In order to determine when to perform placement, one can re-use the criteria used to perform initial placement, find more or even use new ones. Some of these criteria could be:

- The location of the nodes processing the children of the partial subscriptions
- The change in the data rate.
- limited local processing resources.
- etc.

Once one has determined the factors that are used to determine whether to update a partial subscription's placement, one can then determine the threshold to be used in order to trigger the adaptation.

A threshold is a value or set of values related to the criteria used, that can be used to determine when one should trigger a partial subscription placement adaptation. Ideally, this threshold should ensure that the sum of the cost of adaptation and the inherent processing cost is less than the previous processing cost for the affected partial subscription(s).

Due to the dynamic environment of MANET, adaptation is crucial for a placement mechanism to achieve its goal of low message overhead and CEP reliability.

3.1.5 Conclusion

Placement mechanism approaches need to balance between low message cost based techniques which yield sub-optimal results and high message cost based techniques which yield optimal results.

One should also notice the fact that the incentives of finding the optimal placement for a partial subscription depends on its degree of selectivity or the ratio between its input and output. For partial subscriptions with a high degree of selectivity, a high message cost based technique might be appropriate as long as the optimal placement for the partial subscription is found. On the other hand, the message overhead related to finding the optimal placement for a partial subscription might not be necessary if the partial subscription's selectivity is too low.

A query complexity is related to the number of partial subscriptions that are extracted from it and processed in a distributed way. As the degree of a user query's complexity increases, it should take more message overhead to place the derived partial subscriptions.

In a dynamic environment, the need for placement adaptation introduces additional message overhead related to finding a new optimal placement for a query chunk. The rate of adaptation and how optimal the new placement is determine the overall performance for the distributed complex event processing. By finding the right adaptation rate and optimal

placement, one can further minimize the message overhead related to distributed complex event processing.

The degree of a user's query complexity heavily influence the message cost related to placement adaptation. This is due to the fact that placing a query chunk on an other node might trigger placement adaptation for other related query chunks. The more query chunks are affected, the more message overhead will be used for placement adaptation.

In such situations where deterministic approaches are not appropriate while approximate solutions are acceptable, heuristic algorithms can be used to try to find near-optimal solution.

3.2 System model

In this section we present models that are used as a foundation and guideline for the design of the distributed placement mechanism.

3.2.1 Data model

This section describe the data model used in the system. As mentioned earlier, the basic set-up includes one or more sensors producing data samples, other network nodes and the CEP detecting event patterns from sensor data against the user's subscription.

The user expresses her interest in the form of a subscription. In order to enable the distributed processing of subscriptions for complex events, the latter need to be divided into partial subscriptions that can be processed independently. Furthermore, as mentioned earlier, an optimal placement must be found for each of the partial subscriptions in order to minimize the cost of processing the user's subscription. In essence, the placement algorithm is faced with s partial subscriptions and n potential network nodes that can process the partial subscriptions. This problem is similar to the task assignment problem which has been found to be NP-Complete [6] [34]. However, [6] showed that the problem can be solved in $O(nm^2)$ if the n tasks are structured as a tree. For this reason, the partial subscriptions will be structured as trees.

In this project, we assume three kinds of network nodes:

1. Application node or sink which receives the user's subscription to a complex event.
2. Data source node connected to a sensor producing data samples.
3. network node which can be any of the above or any other node from the network.

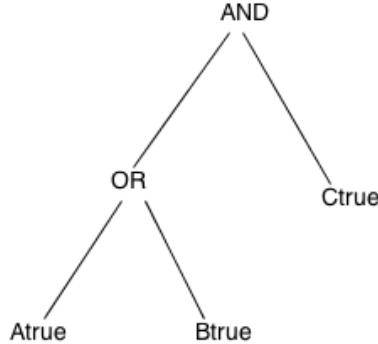


Figure 3.1: a subscription tree

The subscription tree can be represented as the graph:

$$T = (\gamma, \varphi) \quad (3.1)$$

where γ is the set of all partial subscriptions in the subscription tree obtained from user subscription S . φ is the set of the links between the partial subscriptions.

For simplicity we assume that each user subscription is split into a binary tree. By making this assumption, we limit the number of events necessary for a node to match a partial subscription, thus making the process simpler and faster. For example, given subscription S for a complex event E , $E = (A \vee B) \wedge C$ and the following subscription tree would be built: see Figure 3.1

Each partial subscription from γ can only match one event and requires at most two events to produce a new composite one. The leafs of the subscription tree or leaf partial subscriptions (Atrue, Btrue and Ctrue) represent the atomic partial subscriptions. The atomic partial subscriptions match sensor data samples and are thus typically placed on the data sources.

We define the set δ , as the set of all events processed and exchanged between partial subscription processors. Furthermore, $\forall x \in \delta$ there exist a set C_x whose members are required by a partial subscription from the subscription tree in order to produce x . For example, in Figure 3.1, events matched by Atrue and Btrue are members of the set C_D . D is matched by OR partial subscription see Figure 3.1

C_x can hold at most two members and σ_x is the size of x .

A subscription's selectivity refers to the ratio between the amount of data input and the number of events produced. More specifically, given a subscription's input IN and its output OUT , the subscription selectivity is IN/OUT

The bigger the value of this ratio, the higher is the subscription's selectivity.

This obviously result in high message overhead related to event data transmission.

As mentioned earlier, a subscription's degree of selectivity is important in determining how much message overhead should be allowed in the quest to find the optimal partial subscription's placement. It might not be necessary to invest a high message cost in finding the optimal placement for a low selectivity partial subscription.

A subscription's complexity indicates the number of partial subscriptions in the subscription tree constructed for the distributed processing. High complexity subscription require more resources to place their underlying partial subscriptions. Furthermore, their initial placement and preceding adaptation might be more difficult especially in mobile environments.

In this project, we assume the following groups or set of events:

1. The data samples produced by the sensors.
2. The intermediate events are produced by CEP engines but do not yet match completely a user subscription.
3. The final events are intermediate events that match a user subscription.

To allow in-network CEP, subscriptions from the users are split into partial subscriptions which are then distributed on real network nodes to be processed independently. However, since the final event corresponding to the user's interest is a correlation between different intermediate and raw events, network nodes processing subscriptions need to exchange both subscription meta data (during partial subscription placement stage) and event meta data (during event routing). Thus, the distributed complex event processing relies on event and subscription meta data in order to place the partial subscriptions and detect complex event patterns that are of interest for the user.

In this project, the partial subscription meta data must provide the following information in order to enable initial and afterwards dynamic placement of the partial subscriptions:

- The subscription's destination. This information is used to indicate where the subscription should be sent for further placement, or just routing.
- The subscription's parent destination. This information is used to indicate where this subscription's parent subscription in the subscription tree has been placed. Consequently, this information indicates where the matched events from this subscription should be sent for further processing.
- Cost information indicating the cost related to processing this subscription on the node currently holding it.

To enable intermediate event routing between processing nodes, the event meta data must provide the following information:

- Event destination, indicating where the intermediate event should be sent for further processing.

Different placement mechanism use different information to place partial subscriptions and route events, thus subscriptions and events meta data used vary between different approaches to partial subscription placement.

The set L is the set of all placement related messages transmitted in the network including partial subscriptions, events and meta data.

3.2.2 Mobility model

When developing algorithms that will be used in Mobile Ad-hoc Networks, one need to model and simulate the environment in which the protocol will be applied. The protocol simulations involve many parameters including a specific mobility model.

A mobility model is designed to represents the network's nodes movement patterns as well as the variation in their location, speed and acceleration through time [5].

Mobility models can be empirical based or synthetic. Synthetic mobility models are more popular due to their simplicity.

The network's nodes movement patterns differ in different application domains and can be classified based on the movements characteristics. These characteristics are themselves based on the assumption that a node's movement is more or less restricted by its own movement history, the neighbouring nodes and its surrounding environment(obstacles) [5]. These characteristics could be:

- Mobility models with temporal dependency based on movement scenarios where a node's movement is dependent to its previous movement patterns. For example, this kind of mobility model might be used to represent the movement patterns of a rescue team moving people or things from ruins to a safe spot.
- Mobility model with spacial dependency based on the movement scenarios where groups of nodes' movement tend to be correlated. For example, in an Emergency and Rescue Mission scenario, rescue personnel might have a correlated movement pattern around the team leader.
- Mobility model with geographical restrictions which use existing real life obstacles like buildings in order to model expected nodes' movement patterns.

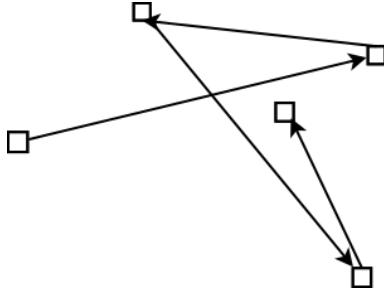


Figure 3.2: This image illustrates a typical node's random movement pattern.

The movement patterns from most application scenarios for MANET are complex and cannot be modelled based on a homogeneous movement pattern. For example, the movement patterns in a rescue mission scenario might exhibit a combination of mobility patterns with geographical restrictions (buildings, ruins, ..), temporal and spacial dependency [14]. Thus, the task of designing a mobility model can be challenging and the resulting model might not be applicable even in other application from the same application domain.

Random mobility models are very popular due to their simplicity. In this project, we assume that while the random mobility models do not really represent any specific real world mobility scenario, they can be good enough to evaluate our placement mechanism performance.

In random mobility models, nodes move randomly. Their speed, direction and velocity are chosen randomly during simulation see Figure 3.2 .

In this project, we use a synthetic based mobility model. Furthermore, for the sake of generality, we use the Random mobility model. More specifically, we use the Random Walk Mobility Model.

Finally, the application node and the data sources are assumed to be static.

3.2.3 Network model

Mobile Ad hoc networks are infra-structureless, self-creating, self organizing and self maintaining. One of the main implications from these characteristics is that the computing devices that form the network must act like end systems and routers at the same time. An other typical characteristic for mobile ad hoc networks is that nodes constantly change their location and thus the inherent network topology is dynamic.

The mobile ad hoc network can be modelled as an undirected graph G built from the set of vertices V connected by edges that make up a set E . The vertices represent the nodes in the MANET while the edges represent the

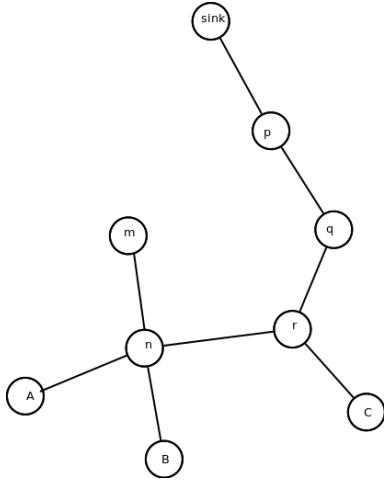


Figure 3.3: MANET with a sink and three data sources for events (A,B and C)

links existing between the nodes in the MANET.

G is the MANET, V is the set of all nodes in G (processing nodes, sink, and data sources), and the set E is the set of all links between the nodes in the MANET.

$\forall i, j \in V, (i, j) \in E \text{ then } (j, i) \in E$. Furthermore, N_i is the set of all nodes adjacent to i .

Consider Figure 3.3:

If the sink wants to send data to node c, nodes p,q,r act as routers for communication between the two nodes.

All nodes have equal transmission and computation power. This assumption does not fairly represent real world scenario for MANET where wireless mobile nodes are typically heterogeneous in their capabilities and capacity. Thus we assume that the computational cost related matching a partial subscription is the same on any node in the network.

Due to the dynamic nature of a typical MANET topology, we assume: $\forall i \in V$, the members of the set N_i will change over the course of time.

In Figure 3.3, the nodes m,n,o,p and q are all processing nodes while the nodes marked with A,B and C are data sources for the respective atomic events (based on Figure 3.1).

3.2.4 Cost model

Part of the overall cost of processing a user's subscription is related to the number of events transmitted inside the network and the number of hops each event has traversed. In particular, in order to process a partial subscription and produce a complex or intermediate event x , a node needs to

get the set of events C_x locally. Thus the cost of processing the partial subscription for x corresponds to the cost of sending events members of C_x from their source to the node responsible of producing x in addition to the computational cost of performing the actual event correlation.

The overall cost related to processing a user's subscription also includes the message overhead related to finding the optimal placement for all partial subscriptions from a user's subscription.

Moreover, the message overhead related to adapting the partial subscription's placement plan is also included in the overall cost of processing a user's subscription.

The overall cost of processing a user's subscription includes:

1. The cost of finding the optimal placement for all partial subscriptions belonging to the set γ . This cost corresponds to the message overhead Υ necessary to find the optimal placement for each partial subscription in the subscription tree.
2. ι is the distance traversed by all events from δ .
3. The cost ζ related to updating the placement plan in order to maintain an optimal execution plan and keep a low message cost for the user subscription processing.

As a result, the overall cost λ of processing a subscription from the user can be described as:

$$\lambda = \Upsilon + \iota + \zeta \quad (3.2)$$

A good placement mechanism will yield minimal ι cost. This would be the result of an optimal placement for all partial events. On the other hand, if the minimal ι cost comes to the expense of a high Υ cost, the result might be poor for the overall λ cost. Furthermore, the ζ cost must also be kept low unless the overall processing cost is increased considerably. To achieve this, the adaptation rate must be set appropriately while the message overhead related to finding the optimal replacement node is kept low.

3.2.5 Formal problem definition

In this project we aim to investigate, design, develop and evaluate a placement mechanism that minimize the overall cost given the following:

- A set of partial subscriptions forming a subscription tree T .
- A mobile ad hoc network G and V the set of all nodes in G .

The main tasks of a placement mechanisms include:

1. To find an optimal placement for each partial subscription belonging to the set γ with minimal Υ cost.

2. To minimize the size of δ and route each event $e \in \delta$ while minimizing the overall cost ι
3. To update the placement plan in order for it to reflect current topology and network state. The cost ζ should be minimized as well.

For each user's subscription, the placement mechanism performs these tasks and the resulting overall cost λ is:

$$\lambda = Y + \iota + \zeta$$

The ultimate goal is to minimize λ cost.

3.3 Alternative one

3.3.1 Initial placement and event routing

This approach explores a pure distributed placement mechanism implementation based on the classical Bellman Ford algorithm. It is inspired by the work done in [34] which is itself an inspiration from the Bellman Ford algorithm.

The mechanism does not assume any network knowledge, thus all the nodes participate in the placement scheme.

The mechanism is scalable in the sense that only neighbouring nodes need to communicate when trying to find the optimal placement for different partial subscriptions.

The partial subscriptions tree obtained from a user subscription is exchanged between neighbours from the application node to the data sources. Upon reception of the partial subscription tree, each node determines the cost of processing each of the partial subscriptions from the tree.

The optimal placement for all partial events is found eventually by neighbouring nodes exchanging cost information based on their own local cost state information and that of their own neighbours.

The mechanism eventually converges when the optimal placement for all partial subscriptions has been found and no additional state information update are available.

The mechanism has two stages:

1. Initialization: A subscription tree obtained from a user's subscription is flooded inside the network. All nodes in the network sets their local processing cost information related to each subscription from the subscription tree. At the end of this stage, all nodes in the network have set local processing information related to each partial subscription in the subscription tree to " ∞ " except for data sources. Data sources set local processing cost information for each partial subscription from the subscription tree to ∞ except for the partial subscription whose attribute corresponds to that of sensor data processed locally. In the latter case, the cost of processing the partial subscription is set to zero. This process ends when all nodes from the

network have set their initial cost state information corresponding to each of the partial subscriptions.

2. Cost information exchange: once the data sources have set the cost information related to producing their respective atomic events, they exchange their updated local processing cost information with their neighbours. The latter update their own processing cost information related to the partial subscriptions whose cost update they received from the data sources, and exchange this information with their own neighbours. This process continues until all nodes in the network are updated and no additional information is exchanged between neighbours.

Every time a node receives updates from its neighbours it updates its current state based on the received update information and exchange its new updated information with its neighbours.

Each partial subscription is placed based on the cost information exchanged between neighbouring nodes. In essence, based on cost information obtained from the neighbours, a node knows which is more suited to process which subscription between itself and its neighbours. As a result, it knows where each intermediate event should be sent for processing.

The main purpose is to construct an overlay network for event routing. Thus, the main goal is to determine whether a partial subscription should be processed locally in order to become the source of its matched events or whether one of the neighbours is better suited to produce the same event while it simply acts as a forwarder for the event towards the neighbour. Determining whether to process a partial subscription p and thus becoming its matched event x data source should be based on how cheap it is to get the events C_x it must match on the local node.

If, based on information exchanged with its neighbour, a node finds out that it is cheaper for it to obtain all the events from C_x than any of its neighbours, than the new cost state information related to the corresponding partial subscription becomes:

For $i \in V$, the total cost $\lambda(i, x)$ to detect event x at node i is:

$$\lambda(i, x) = \rho(i, x) + \sum_{a \in C_x} \lambda(i, a) \quad (3.3)$$

In this case, the node i is the processor for the partial subscription that matches event x . This state information is exchanged with the neighbors such that whenever one of them gets one of the events in C_x it forwards them to i .

If, on the other hand, a node finds out that it is cheaper for one of its neighbours j to get events from C_x necessary for matching x , it sets the local cost state related to partial subscription used to match x as:

$$\lambda(i, x) = \lambda(j, x) + \sigma_x \tau(j, i) \text{ where } j \in N_i. \quad (3.4)$$

In this case, the node i should always forward events from the set C_x to the neighbour j .

Each node in the network that receives an event in the set δ uses the state information to know the next hop for any $x \in \delta$.

3.3.2 Placement adaptation

As mentioned earlier, when a node has new updated state information, it exchanges it with the neighbours. Thus, any change in the network that impacts the state information will trigger state message exchange between neighbours.

3.4 Alternative two

3.4.1 Initial placement and event routing

As mentioned earlier, distributed approaches for placement mechanism come with a high message complexity and might take long to find the optimal placement for partial subscriptions since all nodes typically participate in the placement scheme.

A high message overhead scheme is only needed for partial subscriptions with high degree of selectivity. Otherwise, a near optimal approach might be a better suited solution as long as it has a low message cost. Similarly, using a high message overhead to find the optimal execution plan based on variable that are constantly changing is not efficient since the high targeted high performance might deteriorate quickly.

On one hand, the distributed approach while accurate is inappropriate for mobile environments. On the other hand, the solution to the optimization problem at hand need only be optimal in some cases. In this situations, heuristic algorithms are often suitable and more appropriately than deterministic ones [19]. A heuristic algorithm is one that either give an approximately right answer or part of instances of the solutions. In the case of our optimization problem, it has been showed that near-optimal solutions are acceptable in cases where partial subscriptions do not have high selectivity degree.

The goal with this approach is to use network information in order to find a near-optimal placement for all partial subscriptions.

We assume knowledge about the network topology even though it might not be consistent throughout the network.

Every node in the network knows the address of all the data sources.

Using knowledge of the network topology and the location of the data source, this approach limits the number of nodes participating in the placement mechanism by only involving those nodes that are on the path towards data sources.

Reducing the number of nodes participating in the mechanism should minimize the amount of message overhead necessary to find the optimal placement for partial subscriptions. Furthermore, this approach could work well with an energy management scheme by letting it switch off those nodes that are not participating in placement process.

To achieve the reduction of nodes participating in the placement scheme, we define a set S whose members represent a subset of all nodes adjacent to a node i . Using route information towards all data sources from node i , the latter only sets a neighbour as member of S if and only if it is on the route to one or more of the data sources. In the end S should contain the least possible number of neighbors through which it can reach all the data sources. The set S is similar to OLSR's MPR (Multi Point Relay) set. In OLSR each node in the network chooses a set of neighbouring nodes (MPR) which it uses to flood control traffic. These neighbors are selected in such a way that they can be used by the selector to flood control information to every destination in the network. This is an efficient way of flooding control messages while limiting data transmission.

We use the same technique in order to not only limit the number of nodes used to forward data but also the part of the network that participate in data forwarding based on the location of the data sources.

No message exchange is needed in order to place a partial subscription. When a node receives a partial subscription tree, for each partial subscription, it has to decide whether to forward it towards the data sources or place it locally.

For every node that is forwarding a partial subscription, the latter is sent to the neighbour through which the partial subscription's data source can be reached.

The decision to place a partial subscription locally is made based on whether its children subscriptions in the partial subscription tree are being sent towards the same neighbour or not. Intuitively, when a node determines that a partial subscription's children cannot be sent through the same neighbour, it means that the current node lies on the shortest path between the partial subscription's children processors.

The overlay network for routing the events is made of the node processing the partial subscriptions. Each node processing a partial subscription knows the address of the node processing the parent of the partial subscription in the partial subscription tree.

3.4.2 Placement adaptation

The adaptation scheme uses network connectivity information in order to determine when a partial subscription should be updated.

Each node processing a partial subscription with children monitors the routes to nodes processing them. If one of the monitored routes changes, the node performs a centralized placement process for the affected partial subscription.

The mechanism uses a predefined threshold in order to determine whether to trigger a remote placement for a partial subscription or keep the latter placed locally.

If the threshold has been reached, the partial subscription is placed on the appropriate remote node.

The new partial subscription processor notifies the nodes processing the children that it is the new processor of the parent partial subscription. This means that, from now on, the events should be sent to the new processor.

The node processing the parent for the partial subscription that was reassigned to a different node does not need to be notified of anything. Instead, each node that receives an event assumes that it is intended to be processed locally. Additionally, whenever it receives an event, it checks its source, if the latter is different than expected, it updates its list of monitored routes. After updating the list of monitored routes it can then check whether it is necessary to place the parent partial subscription of a different node. If necessary, remote placement is performed accordingly and the same continues until the no placement adaptation is needed.

In order to avoid inconsistent views of the execution plan between the children, the old processors keep forwarding any subsequent event from any of the previous processors of the children partial subscriptions to the new processor. This should not take long since both children processors eventually receive placement adaptation notifications from the new processor.

3.5 Issues and challenges

3.5.1 Alternative one

This approach has a high message complexity due to the fact that all nodes are participating in the scheme and there is no knowledge about the network topology. Furthermore, the fact that each node exchange its own state information with its neighbors every time the state changes, means that in a dynamic topology, there will be a flood of state information message exchange. This can highly deteriorate the performance of the placement scheme.

As mentioned earlier, this algorithm is based on one that was developed for a static network where nodes can fail and come up online again. However, in a dynamic topology, nodes are likely to have different neighbors in different epochs. Thus, the algorithm would have to be extended in order for it to work in such an environment. Furthermore, the state information for each node is only relevant as long as the node is not moving or no new neighbors are appearing. Considering the fact that node movement is the main characteristic of the current network scenario, the mechanism will most likely not work appropriately.

One last observation is that each node that receives an event forwards is based on the state information stored locally. Since the network environment is highly dynamic, the state information used for routing the events will be constantly changing thus making an unstable event routing overlay. Even if events make it through to the destinations, this might be achieved with a high delay.

3.5.2 Alternative two

Typical for heuristic algorithms, this mechanism does not yield optimal solutions to the optimization problem. Thus, in cases where the mechanism is placing highly selective partial subscriptions, the resulting plan might not have good performance.

The near-optimal placement for a partial subscription is found without any message exchange, thus at the lowest message cost possible. However, the fact that the decision for placement is made based on one node's view of the network topology means that the scheme is vulnerable to network partitioning.

3.5.3 Conclusion

Unless extended, the distributed mechanism cannot work as it is in a dynamic environment like MANET. Furthermore, the algorithm is already too complex to implement and extending it would be even more complicated due to the dynamic nature of MANET.

The heuristic approach does not provide optimal solutions to the optimization problem at hand. Furthermore, it is not robust against network partitioning thus in cases where the network is highly partitioned, the mechanism will produce sub-optimal placement plans. On the other hand the minimal message cost and especially during placement plan adaptation compared to the distributed approach might bring the overall cost of processing a user's subscription significantly low and acceptable.

As mentioned earlier, the main purpose of a placement mechanism is to minimize message transmission in the network and thus save energy resources. As mentioned earlier, the overall cost of processing a user's subscription includes:

1. The cost related to finding the optimal placement for all the partial subscriptions obtained from the user's subscription
2. The amount of event exchanged between processing nodes and their respective hop count number.
3. The cost related to updating the placement plan.

Thus, minimizing the overall message cost for processing a user's subscription entails not only minimizing the amount of event sent and their respective hop count number through optimal placement of the partial subscriptions, but also to keeping low the message cost related to finding and updating the optimal placement of partial subscriptions.

The heuristic approach seems to minimize the first and last message cost while allowing acceptable or even near-optimal message cost for the second cost related to event routing. On the other hand, the distributed approach promises to yield a minimal cost related to event routing, but this is cancelled by a high message cost related to finding and updating the optimal placement for partial subscriptions.

The heuristic approach seem more appropriate and will be further explored and implemented in this project.

3.6 Heuristic based distributed placement mechanism

As mentioned earlier, the mechanism must first find the optimal placement for each partial subscription, building stepwise an event routing overlay. Additionally, it must appropriately forward intermediate events between partial subscription processor nodes and the sink for the final event.

In this section, we first briefly describe the distributed complex event processing middleware for which the placement mechanism was developed. Then, we present a more detailed presentation of the heuristic distributed placement mechanism design decisions.

3.6.1 The DCEP middleware

DCEP middleware architectural design

The DCEP middleware developed in [17] has eight different components that collaborate to enable the distributed complex event processing. CommonSens CEP engine is used as the CEP component in the DCEP middleware.

The following list is a presentation of each of the DCEP middleware components along with a brief description for each.

- The communication component is responsible of receiving and sending data from and towards the user application and other remote nodes running the middleware. It can also provide cross layer information about the network topology.
- The splitting component is responsible of splitting the user subscription into partial subscriptions that can be processed in a distributed way.
- The dispatcher component is responsible of forwarding messages between the local middleware components.
- The resource manager component has information about resources availability. One of the key information provided by resource manager and used in the placement mechanism is the location of data sources.
- The activation and deactivation component deactivate or activate partial subscriptions based on whether there are resources available on the node.
- The data store holds partial subscriptions and data tuples in memory, making them available for later retrieval.
- The placement component is responsible of placing partial subscriptions and data tuples for distributed processing inside the network.

Placement mechanisms are used by the placement component in order to perform its task. The placement component uses different placement mechanisms according to the current system configuration. The middleware is always run with one of the available placement mechanism.

The main purpose of the placement component is to determine where a partial subscription or event should be placed for processing (locally or on a remote node).

Different placement policies and mechanisms can be used to achieve different goals in terms of various performance metrics. For example, the distributed placement mechanism developed in this project aims to reduce the number of messages transmitted during complex event processing.

Different policies might be used in order to take advantage or deal with a specific environment or resources. For example, some placement mechanisms developed in [17] use mobility (network ferries) in order to handle network partitioning.

The placement component is a front end for different placement mechanisms. The other components do not need to know which placement mechanism is currently being used to perform placement.

Moreover, the placement component provides generic functions that are not specific to any placement policy.

The placement mechanism needs to perform two main tasks:

1. Subscription placement: determining where partial subscriptions should be placed for processing.
2. Event placement: determining where events should be sent for further processing or delivery to the user application.
3. Adaptation

For this thesis the placement mechanism also needs to adapt the execution plan if necessary.

After the initial subscription placement, the placement overlay looks like the one in Figure 3.4. In that figure, the user subscription could be $E = C \vee (A \wedge B)$, where A, B and C are produced respectively by data sources s1, s2 and s3. In this placement overlay, $(A \wedge B)$ would be matched at node s2 and the result would be sent to the sink where $C \vee (A \wedge B)$ would be matched.

The events are forwarded between nodes processing partial subscriptions towards the application node.

Whenever, a route between a node processing a partial subscription and the node(s) processing its children changes, adaptation is triggered. A threshold is used to determine whether to place the parent partial subscription on a different node or keep it placed locally.

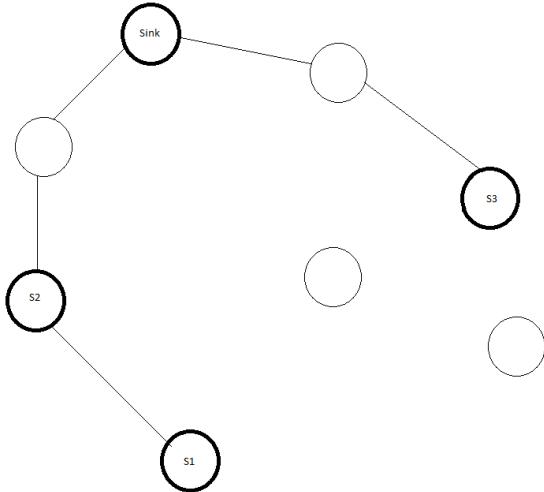


Figure 3.4: A placement overlay network after initial placement

3.6.2 Subscription placement

The following outlines the heuristic-based distributed placement algorithm:

Partial subscriptions are processed and forwarded in bulks during initial placement. Thus, the dispatcher component always waits until the entire bulk of related partial subscriptions is received before forwarding it to the placement component.

When the distributed placement mechanism receives a list of partial subscriptions, it performs the steps described in algorithm 1.

The relay neighbours mentioned in the algorithm at line 6, are used to limit the number of nodes in the network that participate in the initial placement. This is achieved using the information obtained from the resource manager component about the location of the data sources. At line 4 all data sources corresponding to the current partial subscriptions in the list are retrieved from the resource manager component.

Additionally, the relay neighbours allows the placement mechanism to use the shortest path routes towards all data sources. We hope to reduce the number of hops necessary to perform the optimal placement of all partial subscriptions.

As it appears, no cost information is used when placing a partial subscription. The decision to perform placement is made by one node based on the assumption that no node further down the path can forward data between the two children without receiving it or sending it through the current node. Consequently, the current node is considered better suited to process the parent partial subscription.

Algorithm 1 distributed heuristic placement algorithm

```
1: receive a list of partial subscriptions for placement.  
2: subsList ≡ list of all partial subscriptions  
3: for all partialsubscriptions ∈ subsList do  
4:   dSources ← get corresponding data source(s)  
5: end for  
6: relayNeighbours ← the least number of nodes that can be used to  
   forward all partial subscriptions in the list, towards their corresponding  
   data sources.  
7: for all partial subscriptions ∈ subsList do  
8:   for all neighbours ∈ relayNeighbours do  
9:     if the current partial subscription being processed has no children  
     then  
10:      if the current node is not a data source for the current partial  
         subscription then  
11:        if the next hop to the partial subscription's data source is the  
          current neighbour under consideration then  
12:          send the partial subscription to the current neighbour  
13:          process next partial subscription  
14:        end if  
15:      else  
16:        place the partial subscription locally  
17:        process next partial subscription  
18:      end if  
19:    else  
20:      the current partial subscription being processed has children  
21:      if the children are being sent through the same neighbour then  
22:        if the children are sent through the current neighbour under  
          consideration then  
23:          send the current partial subscription to the current neigh-  
            bour under consideration  
24:          process next partial subscription  
25:        end if  
26:      else  
27:        place the partial subscription locally  
28:        process next partial subscription  
29:      end if  
30:    end if  
31:  end for  
32: end for  
33: return 1
```

No information is exchanged between nodes according to who is most suitable to perform placement.

This is an attempt to reduce as much as possible, the message overhead required to perform placement of each partial subscription. Furthermore, we want to make it a simple process in order to avoid complications that might interrupt the whole process.

However, the downside of this approach is that the mechanism might not find the optimal placement for the partial subscriptions.

This design decision is made in order to make the mechanism robust enough for the sudden and dynamic movements of nodes.

The algorithm gets as input:

- A list of partial subscription.
- A list where to store information about where each of the partial subscriptions in the list should be sent.

At line 4 all data sources related to the partial subscriptions from the provided list are retrieved.

At line 6 the relay neighbours are retrieved based on the location of the retrieved data sources.

From line 7, each partial subscription is processed based on which node will receive it or its children.

The decision on whether to place a subscription locally or forward it is based on the following cases:

- The first case is one for which a partial subscription has no children (line 9). In this case, the partial subscription should be placed locally if the current node is its data source (line 10) Otherwise it is added to the list of partial subscriptions that are sent to the appropriate neighbour (line 11).
- The second one represent the case where the current partial subscription has children (line 20). In this case, if its children are sent through the same neighbour, the partial subscription is also added to the list of the partial subscriptions that are sent through that neighbour (line 22). If, however, the children are being forwarded through different neighbors or one of them is placed locally, the partial subscription is placed locally (line 27).

3.6.3 Event placement

The event placement task uses placement mechanism meta data created during the initial partial subscription placement stage.

The following algorithm outlines the event placement process:

When a partial subscription is placed locally, corresponding meta data information is stored. Among the information stored in a partial subscrip-

Algorithm 2 Event placement algorithm

```
1: if the event is from the local CEP engine then
2:   send it to the node processing the parent of the partial subscription
      that was used to produce the event
3: else if the event is received from a remote node then
4:   send the event to the local CEP engine
5: else
6:   send the data tuple to the local CEP engine
7: end if
```

tion's meta data is the location of the node processing its parent. This information is used by the event placement scheme in order to determine where to send an event that is produced by locally stored partial subscriptions.

In the case of data sources that are connected to sensor nodes, the latter send data samples to the middleware's communication component which forwards it to the placement component.

The sensor data samples are placed on the local node since their corresponding partial subscriptions (leaves in the partial subscription tree) are always placed on the corresponding data sources.

Since each node knows the other nodes processing parents for the partial subscriptions that are placed locally, events are always sent directly to their location.

Consequently, whenever an event is received on a local node, it is sent to the local CEP engine.

Included in the mechanism meta data is the location of the node processing the parent partial subscription of the locally stored partial subscription. This location is where the events matched by the current partial subscription should be sent.

3.6.4 Adaptation

The adaptation scheme has three main parts:

- placement,
- sending placement adaptation notifications to the nodes processing the children of the newly placed partial subscription, and
- inconsistent execution plan view management.

Placement

The adaptation scheme uses cross layer information in order to trigger placement re-evaluation. This is achieved through a call back function that

is executed every time a route to a destination is changed. The mechanism checks whether the route change concerns a node that is processing one of the children of a partial subscription placed locally. If it is the case, the placement adaptation algorithm is triggered.

The placement algorithm is as follows:

Algorithm 3 Placement adaptation algorithm

Require: the route to one of the nodes processing a child of a partial subscription placed locally has changed

Require: the new route is longer than previous

Require: the location of the nodes processing the children is known

```

1: print child1-processoraddress
2: print child2-processoraddress
3: print parent-subscription
4: route1  $\leftarrow$  route to child1-processor-address
5: route2  $\leftarrow$  route to child2-processor-address
6: new-candidate  $\leftarrow$  last common hop between route1 and route2
7: print threshold
8: if (distance between the current node and new-candidate)  $\geq$  threshold then
9:   place parentsubscription to newcandidate
10: end if
```

For each node processing a partial subscription whose parent is placed locally, a record about the following information is kept:

- The id of the child partial subscription,
- the address of the node processing the child partial subscription.,
- the number of hops from the current node processing the partial subscription to the node processing the child partial subscription,
- the parent partial subscription, and
- the node processing the parent partial subscription.

Every time a partial subscription is placed locally, a data structure containing the information in the list above is created for each child of the partial subscription.

However, at that time not all the information is available to fill in the data structure. Thus, the data structure is filled with appropriate information in two stages:

1. The first stage is when the data structure is created and the subscription IDs of the child and the parent partial subscription placed locally are filled in. Additionally, the address of the current node is filled in. This happens when a partial subscription with children is placed locally, during initial placement.

2. The second stage is during event routing, when the first event from the child is received. The address of the node processing the child is retrieved and filled in the data structure, as well as the number of hops to that address.

There is a special case when a partial subscription is placed locally during placement adaptation. In this case, the data structures for each child are created and filled with appropriate information in only one stage at the reception of the partial subscription to the new processor node.

The Algorithm 3 uses the information in the data structure above in order to determine whether to place a partial subscription on a remote node. Basically, the longest common route for the two nodes processing the children is determined (line 6). Afterwards, the route's number of hops is compared to a predefined threshold in order to determine whether a remote placement should be performed. If the number of hops is greater or equal to the threshold, the affected partial subscription is placed on the last node on the common route towards its children processors (line 9).

Placement update notification

The new chosen processor for a partial subscription needs to place the partial subscription locally and send a notification to the nodes processing the children.

First of all, the scheme used to perform placement adaptation is not the same as the one used to perform the initial partial subscription placement. During initial placement, when a node receives a list of partial subscription, it needs to decide whether to forward or place locally each one of them. On the other hand, during placement adaptation, if a node received a list of partial subscriptions, it means that they have been already placed locally. Consequently, the placement component needs to be able to differentiate between initial placement and placement adaptation. To achieve this, a field member is added to the mechanism meta data type. This field could be a boolean type that is true when the placement overlay message is an update or false if it is part of an initial placement scheme.

Secondly, the new processor needs to know the addresses of the nodes processing the children partial subscriptions. This is also achieved by adding to other fields in the mechanism meta data that represent the two addresses.

The new processor uses a notification message in order to notify the nodes processing children of its locally placed partial subscription that it is the new processor of their partial subscriptions' parent.

When a placement adaptation notification is received by a node, the latter needs to determine the partial subscription placed locally whose parent has

been placed on a different node.

The placement adaptation notification needs to provide this information. However, the message load needs to be as small as possible. Thus, the adaptation notification in this scheme has only one field which contains the previous processor.

At the remote node, the information contained in the notification message is used to determine the local partial subscription whose meta data should be updated.

This is achieved using another local data structure with the following information:

- The subscription id of partial subscription stored locally
- The address of the node processing the parent of the local partial subscription

This data structure is created every time a node is placed locally, using the partial subscription's meta data.

When a node retrieves the subscription ID corresponding to the previous parent partial subscription processor, it uses the subscription ID to get the right meta data

Both partial subscription's meta data are updated accordingly.

Chapter 4

Implementation

4.1 Introduction

The middleware for DCEP [17] has a placement component which determines where partial subscriptions that are obtained from a user subscription should be placed for processing.

This task is crucial for the reliability of the DCEP scheme. Furthermore, we have seen that if optimal placement for each partial subscription is found, it can reduce considerably, the amount of data transmitted in the network and thus save network resources.

The placement component in the middleware is able to use different policies in order to perform placement. At any point in time, the placement component is configured to use one specific policy in order to perform placement.

Different policies might be meant to deal with different issues and thus are suitable in different situations.

The placement component's policies are implemented as placement mechanism modules that can be used to perform placement.

Our heuristic distributed placement mechanism has been developed in order to limit the message overhead while ensuring DCEP reliability.

The notion of an event and a data tuple will be used interchangeably in this chapter.

In what follows, we start by describing the Distributed Complex Event Processing middleware developed by [17]. In this section, the main components are described in terms of how they support the placement mechanism in accomplishing its tasks.

The following section delves into the details of the distributed placement implementation.

4.2 The distribute complex event processing middleware

As mentioned earlier, a middleware was developed by [17] in order to enable in-network complex event processing. Different components were developed as part of the middleware in order to deal with various challenges related to complex event processing in MANET. The placement component was developed by [17] in order to deal with the need to find where related partial subscriptions from a user should be placed for independent processing. Due to the large number of issues related to the task of placement, different placement mechanism were developed.

This thesis' aim is to develop a distributed placement mechanism which minimize the message overhead related to complex event processing while keeping the need for CEP reliability into perspective. However, centralized placement mechanisms were developed as part of the middleware by [17].

The communication component is used to forward messages between nodes processing partial subscriptions. Additionally, it provided cross-layer information necessary for other components' operations (the placement component for example). To send a message to a remote node the function **send_message** is used. The communication component uses the currently configured routing protocol in order to deliver the message to its destination.

Different cross-layer information provided by the communication component are: the route to a specific destination, notification when a route is removed or added, the address of the current node, etc. The function **find** is used to retrieve the route to the destination specified in the function's argument. A route has type **TFullRoute** with information about each node on the route, when the route was detected and whether it is removed or not. This information is used by the distributed mechanism both to make a decision on where to place a partial subscription and find the next hop to a given destination. The current node address is used for addressing on over the placement overlay.

Another important component for the placement mechanism is the resource manager which provides information about where the data sources are located. This information is used to determine where the partial subscriptions should be directed in order to reduce the amount of nodes included in the initial subscription placement, thus minimizing the message overhead.

4.3 Placement mechanism implementation overview

The placement mechanism's main tasks are to:

- Perform initial placement of partial subscriptions in order to enable in-network CEP. As mentioned earlier, the placement mechanism should find the optimal placement for each partial subscription in the subscription tree.
- Perform event routing between nodes processing related partial subscriptions in order to detect complex events of interest to the user.
- Perform placement adaptation in order to counter the effects of network change throughout the course of event processing. As mentioned earlier, without adaptation, the initial execution plan might become inefficient over time due to the dynamic topology among other things.

A placement mechanism class is used to implement the placement mechanism concept. The main functions are:

- **subscription_check_policies** which uses the placement algorithm to find the placement for each partial subscription in the provided subscription tree.
- **data_tuple_check_policies** which determine where to send an event based on stored meta data about partial subscriptions.

Every time a subscription or partial subscription tree is received by the communication component, two main situations are possible:

- The communication component receives a subscription from the user in which case the subscription needs to be split before being sent to the placement component for processing.
- The communication component receives a list of partial subscription in which case they are all forwarded to the placement component.

When the placement component receives a list of partial subscriptions to process, it calls the **subscription_check_policies** function on the currently configured placement mechanism's object.

When the placement mechanism returns, a list of the destinations for each partial subscription is made available for the dispatcher to know which ones must be sent on remote nodes and which ones must be sent to the local CEP engine.

Every time the communication component receives an event to forward, it passes it to the placement component through the dispatcher component. When the placement component receives an event it calls the **datatuple_check_policies** function of the currently instantiated placement mechanism's object.

As mentioned earlier, the adaptation uses the number of hops between a parent partial subscription processor and partial subscription's children processors. This requires cross layer (network layer) information which is provided by the communication component.

To achieve this, the distributed placement mechanism has a differed call-back function which is passed to the communication component so that the latter always execute the placement mechanism function every time there is a route change event.

The callback function from the mechanism is ***monitor_routes***. If the route change concerns a node processing a child partial subscription for a partial subscription that is placed locally, the adaptation scheme is triggered.

To start the adaptation scheme, the function ***check_placement_adaptation*** is called.

The following sections are organized as follows:

- We describe the data structures used to perform placement related tasks
- We describe the message types used to exchange data between nodes processing partial subscriptions
- We describe in more details how each one of the placement related tasks are implemented

4.3.1 Placement mechanism meta data

In order to perform placement, the distributed placment mechanism relies on the following main data structures created in this thesis:

- Subscription meta data ***TSubscriptionMech_Distr*** which includes information about:
 - The ID of the placement mechanism
 - Where the partial subscription should be placed
 - The address of the node processing the parent of this partial subscription. This address is where events produced by this partial subscription should be sent.
 - The node that placed the partial subscription on this node
 - The addresses of the nodes processing the children of the partial subscription. This information is used when a newly selected processor for this partial subscription wants to send an placement update notification to the nodes processing this partial subscription's children.
 - Whether or not this partial subscription is being placed by the placement adaptation scheme. This information is used by the placement mechanism to determine whether the partial

subscription needs to be placed or has already been placed as part of a placement adaptation scheme. Consequently, The same function ***subscription_check_policies*** is used for both initial and adaptation placement schemes. This information is implemented as a bool type which is set to true if the partial subscription is sent during placement adaptation and false otherwise.

- Event meta data ***TDataTupleMech_Distr*** which includes the following information:
 - The ID of the placement mechanism
 - The destination where the event should be sent
- Placement adaptation meta data ***TChild_route_metadata*** which is created for each locally placed partial subscription's children, includes the following information:
 - The partial subscription's ID representing a child of a partial subscription placed locally.
 - The node processing this partial subscription.
 - The partial subscription's parent
 - The node processing the parent partial subscription (it is supposed to be the current node unless the parent partial subscription has been placed on an other node by the adaptation scheme).
 - The number of hops between the current node and the node processing this partial subscription.
- Partial subscription and parent processor mapping data structure ***TSubAndTupleReceiver*** used to store a mapping between every partial subscription placed locally and the address of the node processing the partial subscription's parent. This information is used when a node receives a message concerning placement adaptation for a parent of a partial subscription placed locally.
- Another important type is the ***TSubscriptionDistr*** provides the following information:
 - The subscription ID
 - The subscription format
 - The name of the event produced by this subscription
 - The subscription tree ID to which this partial subscription belongs.
 - The id of this partial subscription in its subscription tree.
 - The number of nodes in the partial subscription tree
 - The number of parent partial subscriptions

- The number of children partial subscriptions.
- The size of the partial subscription expression
- The IDs for the parent partial subscriptions
- The IDs for the children partial subscriptions
- The partial subscription expression itself.

4.3.2 Overlay message types

The placement mechanism uses three messages for partial subscription initial placement, event routing and placement adaptation. These are extended from the ones previously developed by [17] except for the update notification message.

- Subscription message which is used during partial subscription placement.
- Event notification message which is used to send matched event to the next node processing the parent partial subscription.
- Update notification message which is sent by the newly elected partial subscription processor to the nodes processing its children partial subscriptions.

Each of these messages has a header structure with the following information:

- The message type
- communication protocol
- The size of the message. When the message is a fragment, this information represent the size of the entire message the fragment belongs to.
- The message source
- The message destination
- The message ID
- The time the payload content was generated

The subscription message ***TSerializedMsgSubscriptionDistr*** contains the following information:

- The partial subscription ***TSubscriptionDistr***
- The partial subscription's meta data ***TSubscriptionMech_Distr***.

The message type ***TSerializedMsgDataTuple*** used to transport an event has the following information:

- Event ID

- The node that produced the event
- The attribute name of the event
- The value of the attribute name
- The sequence number of the event
- The event's meta data (***TDataTupleMech_Distr***)

Finally, the placement adaptation notification message ***TSerializedPlacementUpdateMsg*** contains the address of the previous partial subscription processor.

4.3.3 Initial placement for partial subscriptions

The communication component might receive a subscription from two sources:

1. A local CEP application
2. A remote node

In the first case, the subscription is sent to the splitting component through the dispatcher and the result is a partial subscription tree which is then sent to the placement component for placement.

In the second case, an attempt is made to retrieve from the data store component the entire tree to which the partial subscription belongs. If the entire tree is found, it is forwarded to the placement component. However, if some of the partial subscriptions that were sent together with the current partial subscription are not yet available in the data store, then the dispatcher waits for the entire tree to be reassembled. The information contained in the ***TSubscriptionDistr*** type is used to determine which partial subscription tree a partial susbcritption belongs to, and how many nodes are in the partial subscription tree.

When the communication component receives a subscription message (***TSerializedMsgSubscriptionDistr***), the partial subscription's meta data contained in the message is stored. Afterwards, the steps mentioned above are taken accordingly.

In our distributed placement scheme, every time the placement mechanism module processes a list of partial subscriptions, they all receive the same tree node id. Furthermore, each partial subscription's meta data contains information about the number of partial subscriptions that are being sent to the same neighbour as itself. This information is later used by the neighbour when it is trying to determine whether all partial subscriptions belonging to the same tree have been received.

As mentioned earlier when the partial subscription tree is re-assembled, the dispatcher calls the placement component. The latter than calls the "***check_policies_subscriptions***" function with the list of partial

subscriptions.

The placement mechanism module relies on the following helper functions when performing placement:

- **get_data_sources** function which simply gets the addresses of data sources related to the current partial subscription tree being processed.
- **get_random_id** function which gets a random number to be used as the subscription tree id for the partial subscriptions being processed.
- **get_nextHopsToDataSources** function which is used to select the least number of one hop neighbour nodes that can be used to forward partial subscriptions to the data sources.
- **check_children_path** function which is used to determine whether the children of a partial subscription will be sent to the same neighbour. The decision on whether to place a partial subscription locally or not is made based on the return value of this function. If this function returns false, the parent of the partial subscriptions that are given to the function is placed locally.
- **set_tuple_receiver** function which sets the current node's address as the event destination for the children of the partial subscription provided as argument.
- **set_tree_nodes_total** function which set the **tree_nodes_total** information for each partial subscription. As mentioned earlier, this information is used by the destination node in order to determine whether the entire subscription tree sent by the current node has been received.

*The **check_policies_subscriptions** function returns a list of objects of type **TSubscriptionDestinations** which holds information about a subscription and a list of its meta data. There is one subscription meta data per destination.*

The dispatcher component uses this information to determine where to send each partial subscription for further processing or final placement.

4.3.4 Event routing

Every time the communication component receives an event message **TSerializedMsgDataTuple** from a remote node, it stores the event's meta data contained in the message and forwards it to the placement component through the dispatcher component.

The placement component forwards the event data tuple to the placement mechanism by calling the **check_policies_data_tuple** function. This function uses the stored subscription meta data in order to determine where the event should be sent.

When the distributed placement mechanism receives an event for placement, three scenarios are considered:

- The event was produced by a partial subscription placed locally. In this case, the placement mechanism uses the stored meta data about the partial subscription which produced the event, in order to determine its destination. The information from the partial subscription meta data used is the address of the node processing the parent partial subscription.
- If the event was sent from a remote node, the placement mechanism assumes the partial subscription that is meant to process it is placed locally. The event is thus sent to the local CEP.
- If the event is a data tuple from a sensor, an event is created and sent to the local CEP.

4.3.5 Placement adaptation

Placement adaptation uses the number of hops between nodes processing partial subscriptions which are related by parent child relationship.

This information is obtained by having a callback function **monitor_routes** which is called by the communication component every time there is a route change.

If the distance between a node processing a partial subscription and another processing the partial subscription child changes, the node processing the parent subscription begins the placement adaptation routine implemented through the function **check_placement_adaptation**.

This function checks whether the new distance is shorter or longer than the previous distance. If it is longer, the other child's data structure used for adaptation is retrieved and the last common hop is retrieved for the nodes processing the two children. If the last common hop is more than three hops away from the current node, the partial subscription is placed on that node.

Before sending the partial subscription to the new processor, a subscription message is created and the field **parent_update** is set to true.

When the placement mechanism receives a partial subscription list with only one partial subscription whose field **parent_update** is set to true, it automatically places it locally without performing the initial placement routine. Furthermore, the address of the nodes processing the partial subscription's children are retrieved from the partial subscription's stored meta data in order to send update notifications to them.

The function **send_parent_update_msgs** is used to send placement update notification messages to the nodes processing the newly placed partial subscription.

Additionally, a new **TChild_route_metadata** data structure is created for

each child of the newly placed partial subscription, and all necessary information are filled in.

If the partial subscription had been placed locally earlier, the parent processor field in both the children's *TChild_route_metadata* data structures is set back to be the address of the current node.

When a placement update notification is received by the communication component, the latter passes it to the dispatcher which forwards it to the placement component by calling the function **handle_update_notification**. The placement component then sends the message to the distributed placement mechanism by calling the function **receive_parent_update_msg**.

When the placement mechanism receives this message, it retrieves the partial subscription whose parent subscription has been placed on a different node. This is achieved using a data structure *TSubAndTupleReceiver* which holds a subscription id and the address of node currently processing the partial subscription's parent.

The stored information from this data structure is updated with the new parent processor's address, in addition to the partial subscription's local meta data.

Whenever an event is received by the placement mechanism for placement, the latter checks whether the *TChild_route_metadata* data structure corresponding to the node that sent the event contains already the address of the event producer. If not, it means that this is the first time an event produced by the specific partial subscription is received on the current node, thus the data structure is updated appropriately.

Additionally, in case the data structure contained already an address for the event producer, if this address is different from the one from which the current event was sent, this means that the child partial subscription placement has been updated.

In this case, the function **check_placement_adaptation** is called in order to determine whether the parent partial subscription should remain placed locally based on the current location of the children partial subscriptions' processors.

If there is another node better suited to process the partial subscription placed locally, the latter is placed on that node before proceeding with placement of the event locally.

We discuss the issue with CEP state management for the partial subscription being placed on a different node in the next section.

4.4 Issues

At any time, a CEP engine will have a specific state which determines what happens when an event arrives. For example, if a partial subscription's expression is $A \wedge B$, the CEP engine might have received A and waiting for B

to do the matching. If the partial subscription is placed on a different node, the new CEP engine at the new processor node will start from a different state where it is waiting for both *A* and *B*. This will obviously lead to one event lost if the state of the CEP engine on the previous processor is not taken into consideration.

Thus, it might be necessary to implement a CEP engine state transfer for the partial subscription that is being placed on a remote node.

Additionaly, there might be cases where adaptation is triggered at more than one node in the network. This situation could lead to incosistencies in the overall execution plan. Moreover, the fact that adaptation is triggered based on local information might lead to a sub-optimal execution plan.

Part III

Evaluation and conclusion

Chapter 5

Evaluation

5.1 Introduction

In this chapter we evaluate the efficiency and reliability of the distributed placement mechanism compared to the existing centralized approach from the work done in [17].

The middleware from [17] runs over a MANET of mobile devices held by rescue personnel, data sources connected to sensors and a control centre where a CEP application periodically sends user subscriptions for complex events in the network. This is happening in the context of a rescue operation being conducted in an area where there has been an earth quake.

One way to evaluate the efficiency of the placement algorithm in this kind of situation would be to recreate the exact scenario in real life and measure the system's performance. While this would be a perfect way to measure the mechanism's performance, it is both an expensive approach and impractical at this stage of development.

An alternative to the approach mentioned above is to use a simulation model as a mean to evaluate the performance or behaviour of the distributed placement mechanism.

Simulation is the process of designing a model of a real system and performing experiments on it in order to either learn more about its behaviour or simply evaluate the system's various operation strategies [15]. The system's various operation strategies mentioned here can be seen as the different alternative processing approach that need to be evaluated in order to determine which one yields better system performance. In our case, we are interested in evaluating and comparing different placement mechanisms in order to determine which one enables more efficient distributed complex event processing.

A system is a collection of entities that act and interact together in order to accomplish a specific goal [21].

In our case, the mobile devices held by rescue personnel, the rescue personnel, the control centre, the sensors, the MANET, the middleware and CEP

applications are entities in our system of interest.

A system's entity or elements has attributes which are characteristics that can be perceived or measured [28]. For example the MANET has attributes like topology, number of nodes, etc. A device has a transmission range, computational power, storage capacity, etc.

A system takes input variables and uses specific operations strategies in order to produce output. The system output can be used to evaluate whether the system operation strategy achieves the predetermined system requirements.

Using the system requirements, one is able to determine quantifiable system outputs that can be used to measure the system operation strategies' performance in relation to the requirements. In this sense, the term system output includes both system output from processing input and the system's effect on its environment. For example, one of the system output taken into consideration could be energy consumption for a system consuming raw material in order to produce a specific product.

In other words a system can be evaluated based on different parameters. A system parameter represents a system entity's characteristic that can be perceived or measured. As a result, different system parameter values might lead to different system output for the same input. For example, reducing the devices' transmission range might produce a different network topology. A different network topology could mean more hops for data transmission, could impact delay and directly reduce or increase the system's efficiency.

Different system operation strategies might lead to different system output using the same input and parameters. As an example in our case, different placement mechanism approaches will produce different output in terms of performance when given the same input and for the same system parameters.

In order to evaluate our placement mechanism, we can compare its output with other placement mechanism based on both input and system parameters. Comparing the different placement mechanisms for the same system parameters and same input will help us determine which approach is more efficient for which system parameters.

If we use various input, we get further insight into how they placement mechanism efficiency differ for different input. This could be important since we might learn that one placement mechanism performs best only for specific input data.

If we compare the placement mechanism for different system parameters, we can learn how the system's output varies for different system parameters for each of the placement mechanism.

As mentioned earlier, part of the process of simulation is to develop a model of the system we want to study. The model is supposed to be a simpler representation of the real life system which helps us better and easily

understand its behaviour and structure.

A model is a representation of the structure and workings of some real world systems of interest [27]. This representation usually captures only those entities that are considered important based on their impact on the predefined quantifiable system output for the system's performance measurement.

The perfect model would incorporate all the important entities of the system being represented while remaining simple enough to be understood and experimented with. A good model is a judicious trade-off between realism and simplicity [27].

Once a model of the system is at hand, simulation can be used to mimic the real life system's behaviour over time using a simulation program.

There are two kinds of simulation tools:

- Discrete event simulation tools for discrete systems: the system's state changes in response to specific discrete events.
- Continuous simulation tools for continuous systems: the system's state changes continuously over time based on predefined equations.

A system can be viewed as continuous or discrete. A continuous system is one whose state changes continuously over time. A discrete system is one whose state changes occurs in finite jumps [29]. A system's state is a collection of variables that are necessary to describe a system at a particular point in time and in relation to the study's objective [21].

The main goal of our system is distributed complex event processing. Furthermore, the events that lead to the detection of a complex event can be seen as a succession of finite quanta. These finite quanta represent the detection of intermediate events. Based on the subscription being processed, there is a predetermined number of intermediate events that will have to be detected before the complex event is finally detected and notified to the user. Thus, one can claim that our system is of discrete nature. Consequently, a discrete simulator is most appropriate for our evaluation endeavour.

In the following section, the initial system requirements are used to identify quantifiable system output (metrics) that can be used to measure the system's operation strategies performance.

The main goal of this evaluation is to compare two centralized approaches (one with distributed processing and another with centralized processing) with our distributed placement mechanism.

The three placement mechanisms represent the different system's operations strategies. This comparison will be done on the basis of the identified quantifiable system output.

The quantifiable system output determines which elements and what attributes the simulation model should focus on in order to make the right

measurements for the targeted system's output.

In Section 3, the performance metrics, input variables and workloads are identified.

In Section 4, the simulation environment are determined: both the tools and system environment.

The last section will address the experiment conditions and settings.

5.2 System model

In this section, the system's requirements are used to determine the evaluation metrics that will be used to measure the performance of our placement mechanism and also be able to compare it with other existing centralized mechanisms.

The metrics are then used to identify the main system elements that are salient for the system's output related to the metrics and thus has an impact on the predefined metrics.

For each system element included in our model, the main attributes that have an impact on the system's output related to the metrics are identified. The system elements attributes are used to investigate different system parameters for the system simulation later on.

The system's input variables are investigated and described.

Finally, the system's elements interaction is briefly described.

5.2.1 Scenario

The scenario represents a situation where there has been an earth quake and a team of rescue mission personnel has been deployed in the disaster area.

The placement mechanism is part of the placement component in the distributed CEP middleware.

The middleware runs over mobile devices held by rescue personnel, data sources connected to sensors and a control centre where a CEP application sends user subscriptions to complex events in the network.

The mobile devices, the data sources and the control centre are connected over a MANET.

5.2.2 System requirements and corresponding metrics

As discussed earlier the system has requirements related to:

- Energy management: Data transmission has been found to be by far the biggest energy consumer and should be kept minimal.
- CEP requirements: CEP offers a near real time event notification service to application domains where there is a need for real time

information. In these application domains, it is important to be notified when an event happens and as quickly as possible.

In order to reduce energy consumption, the amount of transmitted data must be kept minimal. In other words, ***the message overhead*** related to partial subscription placement should be minimized. Furthermore, we must limit the amount of events transmitted and the number of hops they traverse, in order to reduce the message overhead related to event routing. To achieve this we must find optimal or near optimal placement for partial subscriptions.

In order to achieve CEP reliability requirements, events that happen should all be notified to the subscriber. Given a complex event E and the set C_E of sensor data that is necessary to produce it, whenever these events are sensed by sensors, an event notification for E should be sent to its subscriber. Considering the fact that sensor data from C_E needs to be processed in a distributed way before E is detected, a lot can happen before events from C_E are appropriately processed and the complex event notification for E is sent to the subscriber. The dynamic environment of MANET makes it even more challenging, especially when an execution plan update is involved. Thus, delay might occur during event routing which can lead to situations where some of the intermediate events (and inevitably the complex event) are not detected. Furthermore, complex event processing deals with real time data which means that the complex event of interest for the user is only relevant for a short time period. For this reason, CEP reliability must be assured through reduced ***event notification delay*** and a high probability that complexity events that happen will be successfully notified to the user. The latter metric is called ***event delivery ratio***.

In this thesis, we consider the probability that a complex event that happen is notified.

We have now determined the different metrics that we use to determine how well the system and our placement mechanism in particular, perform in relation to the predefined system requirements.

5.2.3 System entities

Based on the identified system metrics, it is possible to determine which system entities should be part of the simulation model.

We need to balance between simplicity and realism. On the one hand, our system's complexity must be reduced in order to be able to understand its operations better and most importantly, focus on what matters the most. On the other hand, an oversimplified model of our system would yield unrealistic results which in turn would cause unrealistic measurements and observations about the real system.

The middleware is a salient element of our system of interest since without it there would be no processing and output of events or messages.

Another crucial element in the system are the user **subscriptions** without which there would be no event processing necessary in the first place.

The actual sensor data is also an important element in the system since events are extracted from them through the system's operations. We refer to this element as **the workload** for the system.

The mobile devices, data sources and the control centre are also important elements of the system since they represent a platform and resource where the middleware and CEP applications can run. For simplicity sake we group all these elements under the same class name **network computing device** which capture there characteristics.

The rescue personnel's mobility makes them an important part of the system's model since they impact the nature of connectivity between the mobile devices and thus the system's behaviour and operations. Most specifically, in this project, we argue that the dynamic nature of the topology caused by the rescue personnel mobility pose additional issues to the task of finding optimal placement and event routing. The dynamic environment means that some of the criteria (hop count between data sources and the current node under consideration for placement) used to determine where to place a partial subscription are constantly changing, which complicates the tasks.

As mentioned earlier, finding the optimal placement for a partial subscription involves a certain message overhead. Furthermore, the location of a partial subscription impacts the amount of event sent and the number of hop count they traverse. Consequently, the location of a partial subscription has an impact on the event notification delay.

The only effects the rescue personnel has on the operation and output of the system is its mobility. Furthermore, mobility has such a huge impact on the system that its characteristics should be further investigated in order to determine which ones has an impact on the system's output and how. This is important for the conception of a valid model that can be used to simulate the operations of the real system and receive realistic output that can help us to learn more about the system performance for different placement mechanisms. Consequently, **mobility** should be considered as an entity of the system model instead of the rescue personnel. This also means that the rescue personnel element can be ignored in the system model since its only attribute that interests us is now an entity of the system.

The network topology is also a crucial element of the system since its state is used to find the optimal placement of partial subscriptions. Thus, it impacts both the message overhead and the event notification delay.

5.2.4 System entities' attributes and models

We have now identified the salient system elements (Subscription, workload, network, mobility) based on their impact to the quantifiable system output that can be used to measure the system's performance in relation to its predefined requirements.

The next step is to determine which attributes of the identified system elements have an impact on the system's output.

Attributes of a system are characteristics of the system's elements that can be perceived and measured [28]. Since these attributes have an impact on the system quantifiable output, the latter will vary based on the attributes values. In other words, the system output can be manipulated by simply changing the values of its elements' attributes. Furthermore, the different system operations strategies performance can be analysed and compared based on the system's output for specific system elements' attribute values.

At any given point in time, the set of the current values pertaining to the identified entities' attributes represent the prevailing system parameter values. In order to compare different system operation strategies, they have to be simulated for exactly the same system parameter values. The system's output from the different operations strategies can then be analysed and compared.

After identifying the different attributes pertaining to each of the system's elements, different sets of system parameter values will be determined. The simulation runs will consist in simulating each placement mechanism for each set of system parameter values for a predetermined number of times (for statistical accuracy).

Network model

The network characteristics have huge impact on the system operations. Our system's network element is a MANET of rescue personnel, control centre and the data sources.

One of the networks characteristics is the communication range of the wireless devices that form the MANET. This characteristic determines the topology of the MANET.

Short communication ranges avoid network congestion due to the fact that network nodes have less neighbours than if they had longer communication ranges. This allows more nodes to send data simultaneously without interfering with each other. Moreover, shorter communication ranges yield more stable communication links between nodes.

However, this might increase the number of hops that an event might traverse before it is processed. A node that would have taken one hop for communication could take two or more hops due to the low communica-

tion range.

Long communication ranges reduce the number of hops between the network nodes, but increases network congestions, hidden exposed terminal issues, etc.

Furthermore, the links between network nodes tend to be unstable the longer the communication range gets.

The main characteristics of the MANET entity mentioned above impact the MANET's topology. The communication range attribute is crucial for our evaluation since the placement mechanism heavily rely on the network topology information (routing information). This makes it an important system parameter.

An other important network characteristic is the number of nodes in the network. We consider the network density as an important system parameter that can impact the placement mechanism performance. Furthermore, the network density is closely related to the network area size. Consequently, an area size system parameter is also necessary for our simulation runs.

Mobility model

The mobility of the processing nodes in the system creates a dynamic topology for the system operations.

The following mobility characteristics have an impact on the system operations output:

- Speed: the speed of the nodes determines how fast the topology is changing. The more speed increases, the more difficult it becomes for network services and the placement mechanism to function appropriately.
- Mobility range: if a node moves in around in a short range of distance, it might even keep the same neighbours and thus it will not impact data processing. However, when the range of movement starts to increase, it alters the network topology. Considering the fact that the placement mechanism's execution plan is based on a specific network topology, the new topology might yield different data traffic patterns and thus reduce the execution plan performance.

Therefore, the speed and mobility range are important parameters for the placement mechanisms evaluation.

The middleware

The middleware is central to the entire evaluation process since it includes the different operation strategies that we need to evaluate.

While the middleware has many characteristics that could be explored and taken into consideration, only the placement mechanism characteristic

matter for this investigation.

As mentioned earlier, we want to compare three placement mechanisms for in-network complex event processing. Thus, the middleware operation strategy attribute or parameter can have three different values corresponding to the three different placement mechanism we want to compare.

5.2.5 System input variables

As mentioned in the introduction, the system output is determined by both the system's entities attributes values and the system input.

Different input values obviously lead to different system output, and the different placement mechanism might perform differently for different input. One mechanism might perform better for specific input and worse in others. Therefore, properly designed input models can help gain an insight in the mechanism performance.

In this section we explore the different system input variables.

Subscriptions

As mentioned earlier, a subscription submitted by a user can be characterized in two ways:

- Selectivity: subscription selectivity represent the ratio between its input and output data.
- Complexity: subscription complexity refers to the number of partial subscriptions that are obtained from it in order to be able perform a distributed event processing scheme.

With this input variable, we want to measure how the selectivity and complexity of a subscription submitted by a user impacts the performance of the different placement mechanism.

For example, centralized placement mechanisms do not use a high message overhead for subscription placement. However, their execution plan might not be optimal due to their reliance on one node's view of the network topology. Consequently, if the subscription submitted by the user has a high complexity and low selectivity, the centralized approach might produce better results than a distributed and dynamic approach which is trying to keep an optimal execution plan at any time.

Workload

The workload is characterized by the number of sensor data tuples sent to the data sources for processing. This represents sensor data samples sent to the data sources. The workload and subscription represent the system input variables.

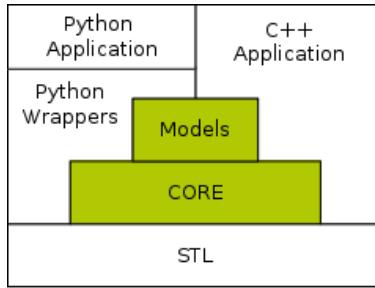


Figure 5.1: ns-3 main components (from www.nsnam.org)

5.2.6 System entities interaction and relationships

The application node, the mobile devices and data sources connected to sensors form a MANET.

A subscription is sent to the middleware running on the application node. The latter splits the subscription and uses the current placement mechanism to determine where the partial subscriptions should be placed for in-network complex event processing.

The placement mechanism uses topology information in order to build an optimal execution plan. The network topology is highly dependent on the current mobility pattern. High mobility might make it difficult to perform distributed placement.

The workload is sent to the data sources which start to process atomic partial subscriptions and produce intermediate events that are then forwarded towards the application node using the event routing overlay which is built during partial subscriptions placement.

5.3 Simulation environment

5.3.1 The tools

In order to use the distributed complex event processing middleware in this evaluation, we use emulation instead of simulation.

NS-3 is a discrete event network simulator for internet systems with emulation capabilities. We use it in this evaluation due to its emulation capabilities (more on this later..) and popularity in the academic world.

NS-3

NS-3 enables simulation configuration, powerful logging capacity, trace collection and analysis. Figure 5.1 shows the ns-3 architectural model.

Ns-3 simulation scripts are written in c++, but simulation scripts written in python are also supported through a python wrapper component which manages access to ns-3 models and core.

c++ or python applications instantiate ns-3 models in order to setup targeted simulation scenarios.

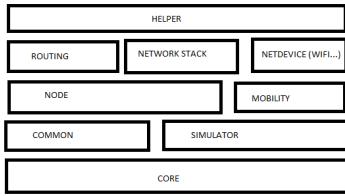


Figure 5.2: ns3 components

Ns-3 has the following main components see Figure 5.2:

- The core component which supports generic aspects of simulation like: logging, tracing, random variables, callbacks, smart pointers.
- The simulator component which supports event scheduling and time arithmetic.
- A common component for objects that are not specific to any network architecture like: packets or tracing objects.
- Mobility component which provides mobility models for MANET.
- Node component which supports fundamental objects for network simulation like: network divides, network nodes, network channels etc.

Different routing protocols(eg. OLSR..) , network stack(eg. IPV4, IPV6, ...) and specific device models (Ethernet, Wifi..) are built on top of the node component. In addition to being able to use these built in models, ns-3 users are able to build their own models from scratch by interacting directly with the node component. They can also extend already existing models.

The helper component contains helper objects that enable users to instantiate corresponding models with default values. This makes it easy to setup simulation scenarios quickly.

Ns-3 supports network emulation as it is able to emit or consume real network packets. As such, ns-3 can emulate a network connectivity between virtual machines. This is a powerful feature that makes it possible to basically emulate communication between computing devices where one can run any application or system software in a real life Linux environment. This is a far better alternative than ns-3 node models. When ns-3 is used to emulate network connection between virtual machines, it creates internal

ghost nodes through which it interacts with the virtual machines.

In this evaluation, we use ns-3 with Linux lxc containers in order to take advantage of a complete real life Linux environment for our mobile nodes. Ns-3 emulates a MANET between the virtual machines using the mobility model of choice.

Linux lxc containers

The lxc container technology (lxc tools) allows the creation of virtual environments (lxc containers) inside a linux host machine. We use this technology to create virtual environments with separate process and network space.

Lxc tools enable ressource management through the control groups (cgroups), and resource isolation through the namespaces.

Control groups provide a mechanism for organizing sets of processes into hierarchical groups and allocating resources (CPU time, system memory etc..) at the group level [20]. Child cgroups inherit certain attributes from their parents. Access to resources can also be restricted at the process group level.

The different namespace features used to enable resource isolation are:

- Network namespace: each lxc container gets its own network stack with a mac address and an ip address.
- PID namespace: lxc tools place lxc containers into a separate PID namespace. The first lxc container created will receive PID number 1 Despite the fact that the host's operating system sees the processes running in each lxc container, their PIDs are appropriately translated to avoid conflicts with real host Operating system PIDs.
- UID namespace: each lxc container gets its own UID namespace.
- Utsname namespace: each lxc container is able to create its own utsname.

5.3.2 Emulation environment setup

As mentioned earlier ns-3 can provide network emulation between virtual machines. In this case, some of the Ns-3 models are replaced by "real world" implementation.

In this evaluation, we replace Ns-3 nodes with lxc containers. This is achieved using the Ns-3 TapBridge Models which integrates real world internet hosts into ns-3 simulations.

In order to be able to integrate real world hosts into ns-3 simulations, they must support TUN/TAP devices. TUN/TAP devices are virtual network kernel devices. A TUN device simulates a network layer device while

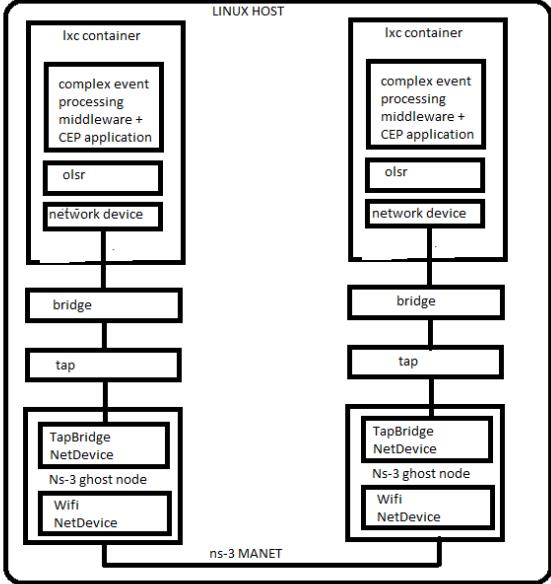


Figure 5.3: simulation environment setup (obtained from: <http://www.nsnam.org/wiki/>)

a TAP devices simulates a link layer device. When a user space program attaches itself to a TAP device for example, it can receive packets that are sent by the operating system through the TAP device. Additionally, the user space application can also send data packets to the operating system's network stack through the same TAP. Consequently, the operating system views these packets as if they were coming from a remote node.

Ns-3 uses TapBridge model in order to attach to a TAP device. The TapBridge model can be configured to configure the TAP itself or use the TAP as it is. In the latter case, the TAP device must be configured in advance before simulation.

Using these feature, we create an emulation environment consisting of both ns-3 and lxc containers see Figure 5.3.

For each node in the emulated MANET, the distributed complex event processing middleware, the CEP application, the CEP engine and olsr are running in an lxc container. Ns-3 emulates a MANET between all the lxc containers created.

The virtual environment provided by LXC has a separate process and network space. The container's network devices are connected to the host Operating system through the linux bridges.

Ns-3 is connected to the linux bridges through a tap device using ns-3 TapBridge NetDevice model. Internally, ns3-uses a ghost node instead of a real

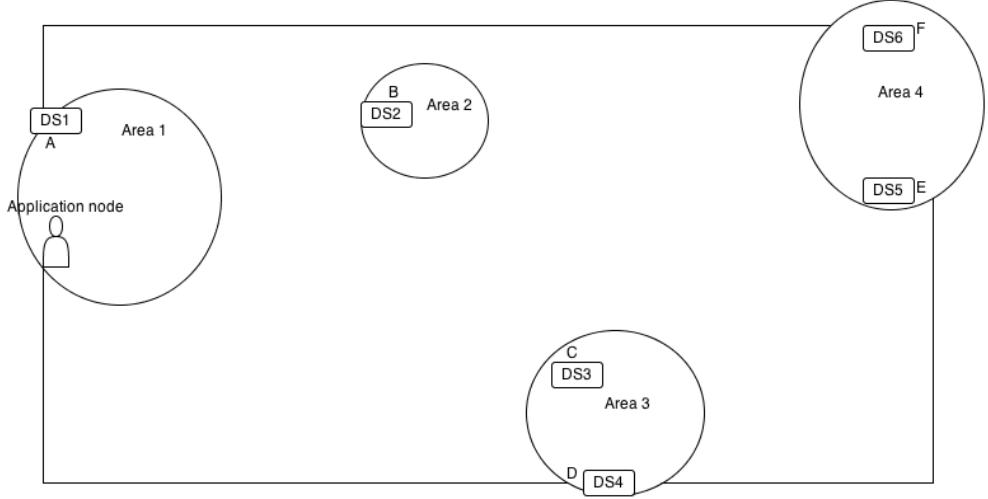


Figure 5.4: The emulation perimeter and data sources location

ns-3 node.

The ns-3 TapBridge NetDevice connect to the tap device and passes packets from the linux container to the internal ns3 ghost node.

The ghost node forwards network packets to the appropriate ns-3 ghost node through the ns-3 wifi NetDevice.

Packets received from an ns-3 wifi NetDevice are forwarded to the tap device connected to the TapBridge NetDevice.

In essence, ns-3 ghost nodes act as lxc container proxies inside ns-3.

This simulation environment setup is based on [17]. The CEP engine used is CommonSens and the routing protocol is Olsr.

5.4 Experiment

5.4.1 Assumptions

We assume a scenario where the search and rescue workers are spread over a an area with $100 \times 60 m^2$ with building ruins. Furthermore, due to the obstacles represented by the building ruins we assume a transmission range of 20m. The data sources are located in the encircled areas in Figure 5.4. In order to avoid network partitions which the placement mechanisms cannot deal with, we use 30 nodes during the emulations. This is due to the fact that networ density with less nodes have led to network partitions. We also do not assume any node failure.

5.4.2 System parameter values

Based on the system elements attributes identified earlier, the following parameter values are used:

- The number of nodes is 30,
- the mobility model is Random Mobility Model, and
- the mobility speed is 0.25m/second.

5.4.3 System input variables

Subscriptions

Early experiments [17] have shown that the selectivity and complexity of subscriptions have a significant impact onto the performance of placement mechanisms. Consequently, we use the following subscriptions with various selectivity and complexity levels:

$$A \vee B \quad (5.1)$$

$$A \rightarrow C \quad (5.2)$$

$$(A \vee B \vee C \vee D) \vee (E \vee F) \quad (5.3)$$

$$A \wedge B \rightarrow C \wedge D \rightarrow E \wedge F \quad (5.4)$$

$$D \vee E \quad (5.5)$$

$$(C \wedge D) \vee (E \wedge F) \quad (5.6)$$

Subscriptions 5.2, 5.1, 5.3 and 5.4, are taken from earlier work by [17] and the two last subscriptions were developed for this evaluation. All these subscriptions should be considered together with Figure 5.4 in order to gain better understanding of the environment setup.

Subscriptions 5.1, 5.3 and 5.5 have a low level of selectivity for both their leaf and root partial subscriptions.

Subscriptions 5.2 and 5.4 have a high selectivity for both the leaf, internal and root partial subscriptions. However, the leaf partial subscriptions are more selective than the internal and root partial subscriptions.

The Subscription 5.6 has low selectivity for the leaf partial subscriptions while the root partial subscription has high selectivity.

Workload

The workload is typically sent to the selected data sources where they are forwarded to the DCEP middleware. The data sources are selected based on the atomic events that are described in the subscription being processed. Figure 5.4 shows which atomic event is produced by which data source. The workload is determined by the type of subscription currently being processed. Subscriptions 5.1 and 5.4 have leaf partial subscriptions with temporal constraints. 20 sensor data samples must be read consecutively from the sensors for the leaf partial subscriptions to match an atomic event. The other Subscriptions have partial subscription leafs that need one sensor data sample to match an atomic event.

5.4.4 Simulation models

We first evaluate the performance of the placement mechanisms for subscriptions with low complexity. This basically represent those subscriptions whose subscription trees have very few levels and typically few partial subscriptions. The targeted subscriptions here are Subscriptions 5.1, 5.2, and 5.5. These subscriptions have various levels of selectivity for the partial subscriptions that are used to detect atomic events. Based on Figure 5.4, Subscriptions 5.2 and 5.1 involve data sources that are relatively close to the sink especially the data source for *A*. Assuming this can have an impact on the results, we developed Subscription 5.5) whose data sources are located far from the sink. This will help us better understand the results for low complexity subscriptions.

The other parameter values are based on the ones provided in Section 5.4.2.

The Subscriptions 5.4, 5.3, and 5.6 have a high level of complexity. Additionally, they vary in their level of selectivity for individual partial subscriptions. The subscriptions 5.4 and 5.3 from [17] are rather homogeneous in that all their underlying partial subscriptions have either low or high selectivity. Thus, we have developed Subscription 5.6 whose underlying partial subscriptions have mixed levels of selectivity.

The other parameter values are based on the ones provided in Section 5.4.2.

5.4.5 Run conditions

A shell script is used to run the entire simulation including ns-3 simulation scripts.

The script receives the following arguments:

1. The mobility scenario to be run which specifies an ns-3 simulation script that should be used to emulate the MANET. In this evaluation, we use the random waypoint mobility model in all runs.
2. The placement mechanism which is an id number representing one of the placement mechanisms under consideration.

3. The partial subscription type which represent one of the subscriptions presented in Section 5.4.3.
4. The buffer size which determines the CEP engines' buffer size. In this evaluation, we use a buffer size 0 which puts a high reliability requirement for the placement mechanisms' delay metric.

First the lxc containers are created using predefined configuration files. Then the appropriate ns-3 simulation scenario is run. Afterwards, the OLSR daemon, CommonSens and the middleware are started on each lxc container.

Afterwards a subscription is sent to the distributed complex event processing middleware of the application node. The latter splits it and sends the resulting subscription tree to the placement component.

When all partial subscriptions have been placed, the workload is sent to the data sources according to the type of subscription being processed. Basically, the workload increases based on the subscription's level of selectivity. Additionally, the number of data sources increases with the subscription's level of complexity.

Log files for the lxc containers, the different services and ns-3 are stored for further analysis.

Each simulation model is run five times for each placement mechanism in order to get statistically valid data.

5.5 Results

In this section, we present and discuss the results obtained from emulating our system with the input variables and parameter values described respectively in Sections 5.4.3 and 5.4.2.

The number of subscription messages represent the recorded number of messages of that type that are used to distribute the subscriptions in the MANET including retransmitted messages. The same applies to the event messages.

Delay is measured starting from the time a sensor reading is received by the middleware for DCEP at the data source to the time the corresponding complex event is received by the middleware. This measurement varies based on which type of workload is currently being processed. For the workload where one sensor data sample is needed to detect an event, the starting time used to measure delay is when a sensor data sample that matches the leaf partial susbscription is received by the middleware for DCEP. For the workload where the leaf partial subscriptions require more than one sensor data sample to match an atomic event, the starting time used to measure the delay is when the last sensor data sample necessary to detect the atomic event is received by the middleware. FOr example, if

30 nodes	constant mobility speed of 0.25m/sec
30 nodes	constant mobility speed of 0.50m/sec
30 nodes	constant mobility speed of 0.75m/sec
30 nodes	constant mobility speed of 0.25m/sec
40 nodes	constant mobility speed of 0.25m/sec
50 nodes	constant mobility speed of 0.25m/sec

Table 5.1: Network scenarios used

a leaf partial subscription requires 20 consecutive sensor data sample in order to detect an atomic event, start time for the delay measurement will correspond to when the 20th sensor data sample is received by the middleware.

Every time a placement related message sent is not acknowledged by the end receiver, the message is retransmitted. The column for retransmissions results contains the combined number of retransmitted event and subscription messages.

We also evaluate the performance of the distributed placement mechanism for various network scenarios in terms of mobility speed and network density. To achieve this we run the emulation first with varying mobility speed. Afterwards, we run the emulation with varying network density by changing the number of nodes parameter value. Thus the distributed placement mechanism is evaluated for the scenarios shown in Table 5.1.

5.5.1 Results for subscriptions with low complexity

The message overhead is determined by the workload for the subscription being processed. The workload is the amount of sensor data samples which is sent to the CEP engine at the data sources for processing.

The results shown in Table 5.2 represent the performance of the centralized mechanism for the centralized CEP scheme. In this scheme, all sensors data samples are sent to the central CEP engine from the data sources.

The workload for the Subscription 5.1 is 12 sensor data samples from the two data sources DS1 and DS2, while Subscription 5.2 processes 204 data samples from the two data sources: DS1 and DS3 see Figure 5.4. This explains the high message overhead for Subscription 5.2 compared to the results for Subscription 5.1.

For Subscription 5.1, the centralized CEP needs only to detect event *A* or *B* in order to match complex event for the subscription. Consequently, the delay for detecting the complex event depends on how far is the location for the closest data source for one of the events required. In fact, as it appears

Centralized processing placement mechanism					
	subscription	event	detected events (probability)	delay (ms)	retransmissions
Subscription 5.1	o	88	1	173	86
Subscription 5.2	o	3318	0.1	296933	503
Subscription 5.5	o	1244	1	109765	411

Table 5.2: Results for centralized processing with the centralized placement mechanism

in Figure 5.4, data source DS1 where sensor data samples for event A are sent from is located in the same area as the sink. Additionally, based on Olsr routing information, the data source DS1 and the sink are one hop away from each other. Consequently, the complex event for Subscription 5.1 is detected with a short delay. This also leads to a higher probability to detect complex events for the subscription.

Subscription 5.2 however, requires all the data samples for each event from both data sources to be received at the sink before the atomic events and the complex event are detected. This means that the central CEP needs to wait for sensor data samples from DS3 before it is able to detect the atomic event C and match it with A which is most likely already detected (considering the location of its data source), in order to detect the complex event. Therefore, the delay related to the detection of the complex events for Subscription 5.2 is higher while the probability to detect complex events is significantly lower than for Subscription 5.1. Another reason for the higher delay registered for Subscription 5.2 is the fact that it takes less time to detect one atomic event for Subscription 5.1 than the time required for atomic events with Subscription 5.2. As an example, it takes 20 consecutive data samples (data samples whose timestamps are within the predetermined time interval) to detect the atomic event C while it takes just one sensor data sample to detect event B. Considering the fact that in each emulation run, sensor data samples are sent in a continuous manner with no pause in between, it is obvious that it will take longer to detect the complex event for Subscription 5.2.

As opposed to Subscription 5.1, Subscription 5.2 has a higher temporal constraints which makes it less likely to detect the complex event. The operator → from Subscription 5.2 means that sensor data samples must not only get to the sink in the entirety, they also have to be processed in a specific order.

Results from Table 5.2 show a higher message overhead for Subscription 5.5 compared to Subscription 5.1 despite the fact that both subscriptions have the same workload size see Table 5.2. This can be explained by the fact that data sources for Subscription 5.5 (DS4 and DS5) are located

Centralized tree placement mechanism					
	subscription	event	detected events (probability)	delay (ms)	retransmissions
Subscription 5.1	43	56	1	283	13
Subscription 5.2	55	204	0.8	341349	90
Subscription 5.5	130	233	1	87489	133

Table 5.3: Results for distributed processing with the centralized placement mechanism

further away from the sink see Figure 5.4. This means higher number of hops between the data sources and the sink, and thus, a higher number of message transmissions between data sources and the sink.

However, unlike Subscription 5.2, complex events for Subscription 5.5 are all detected due to its much lower spatial and temporal constraints.

Results from Table 5.3 show a significant message overhead reduction due to the fact that sensor data samples are now processed by local CEP engines at the data sources. This represents a significant message overhead reduction especially for Subscription 5.2 whose leaf partial subscriptions have a high selectivity. Consequently, results from Table 5.3 show a higher probability to detect complex events for Subscription 5.2. This is due to the fact that in the centralized scheme, the complex event detection relies not only on events *A* and *C* being processed in the right order, but for each event *A* or *C*, all its sensor data samples must be received and processed in the right order. The latter condition is harder to achieve due to the dynamic nature of MANET topology which leads to delays and out of order delivery of the sensor data samples. For example, if one sensor data sample takes a lot longer than the others from the same time interval to reach the sink, the event *C* corresponding to that particular time interval will not be detected. As a result, the complex event will be missed.

However, with the centralized tree placement mechanism for distributed CEP, events *A* and *C* are detected to the data sources. This means that instead of around 40 messages (44 sensor data samples from the two data sources) having to be delivered in time and in the right order to the sink in order to detect a complex event, only 2 messages (events *A* and *C*) need to be received and processed in the right order at the sink.

Subscription 5.5 has a significantly lower delay than the Subscription 5.2 due to its lower temporal and spatial constraints. The complex event will be detected whenever the event from the closest data source is received at the node processing the top level partial subscription for Subscription 5.5. With the Centralized tree placement mechanism, the node processing the

Distributed placement mechanism					
	subscription	event	detected events (probability)	delay (ms)	retransmissions
Subscription 5.1	12	54	1	195	9
Subscription 5.2	27	178	0.8	172244	89
Subscription 5.5	88	192	1	101307	133

Table 5.4: Results for high complexity subscriptions with the distributed placement mechanism

top level partial subscription happens to be 2 hops away (most of the time considering information from Olsr) from DS4. Consequently, the complex events for Subscription 5.5 are detected with a short delay. However, unlike the case for Subscription 5.1 where the node processing the top level partial subscription is the sink, the complex events detected for Subscription 5.5 must be sent over to the sink which introduces additional delay.

The partial subscriptions for Subscription 5.1 and Subscription 5.2 are placed on the same nodes for both the placement mechanism schemes for distributed CEP, thus the results from Tables 5.3 and 5.4 are more or less the same. For Subscription 5.5, the distributed placement scheme places the top level partial subscription at a slightly better location in the network. More specifically, based on the log information from the two data sources in both placement mechanisms for distributed CEP, DS4 and DS5 are respectively 3 hops and 4 hops (sometimes 2 hops) away from the processor of the top level partial subscription in the distributed placement scheme. However, for the centralized scheme, DS4 and DS5 are respectively located 2 hops and 9 hops away from the node processing the top level partial subscription. This can explain both the lower message overhead during event routing for the distributed scheme and the lower delay for complex event detection for the centralized scheme. The lower message overhead for the distributed scheme is due to the fact that the events from the data sources are transmitted over fewer hops.

For all subscriptions, the message overhead related to routing the partial subscriptions is significantly lower for the distributed placement scheme. This is due to the fact that in the distributed scheme, the subscriptions are forwarded in a hop by hop manner between placement component from neighbouring nodes. This means that nodes that are forwarding subscriptions are able to pick shorter routes towards the data sources at the last moment. Shorter routes lead to fewer subscription messages transmission. Additionally, because subscriptions are sent between neighbours,

Centralized processing placement mechanism					
	subscription	event	detected events (probability)	delay (ms)	retransmissions
Subscription 5.3	0	2678	1	352	795
Subscription 5.4	0	75929	0	0	36632
Subscription 5.6	0	3130	0.7	111650	1131

Table 5.5: Results for centralized processing with the centralized placement mechanism

fewer data retransmissions are used since messages are acknowledged more quickly by neighbours.

5.5.2 Results for subscriptions with high complexity

The subscriptions considered in this section have more partial subscriptions and more data sources involved. In the Centralized CEP scheme this leads to higher message overhead compared to lower complexity subscriptions with which fewer data sources are involved. This can be seen from the results shown in Table 5.5.

Subscription 5.4 can be seen as a extreme case of Subscription 5.2. Subscription 5.4 has the same workload for each data source compared to Subscription 5.2. However, Subscription 5.4 has six data sources instead of just two for Subscription 5.2. Consequently, Subscription 5.4 has a significantly higher message overhead. Moreover, no complex events are detected. The high amount of message retransmissions from Table 5.5 suggests a possible network congestion which leads to higher delay for sensor data samples delivery at the sink and more complex events missed due to the subscription's temporal and spatial constraints.

Compared to Subscription 5.5 in Table 5.2, Subscription 5.6 has 2 more data sources. However, the workload for the data sources is the same for the two subscriptions. Consequently, the difference in message overhead is lower compare to Subscription 5.2 and Subscription 5.4. Moreover, due to more constraints for the top level partial subscription, Subscription 5.6 has a lower rate of complex event detection than Subscription 5.5.

Similar to previous observations for Subscription 5.1 and 5.2, the centralized placement scheme for distributed CEP yields a significant reduction in message overhead for Subscription 5.3 and Subscription 5.4. A high amount of sensor data samples is now processed locally at the data sources.

Centralized tree placement mechanism					
	subscription	event	detected events (probability)	delay (ms)	retransmissions
Subscription 5.3	754	1360	1	555	1116
Subscription 5.4	734	1000	0.48	853597	759
Subscription 5.6	777	36	0.3	89287	276

Table 5.6: Results for distributed processing with the centralized placement mechanism

Additionally, for Subscription 5.4, the partial subscriptions placed inside the network are highly selective which leads to fewer events being sent over the network to the sink.

Subscription 5.4 has lower probability of complex event detection compared to Subscription 5.3. This is due to the fact that the partial subscriptions Subscription 5.4 have more constraints than partial subscriptions for Subscription 5.2. More specifically, the complex event for Subscription 5.3 is detected as soon as event *A* is received by the CEP engine at the sink. It is not dependant on the delays related to event from other data sources and the delay related to intermediate event processing at nodes processing internal partial subscriptions. However, all these constraints apply to the detection of the complex events for Subscription 5.4, which explains the low probability for complex event detection see results in Table 5.6.

The high constraints of the internal partial subscriptions for Subscription 5.6 and their high selectivity limits the number of intermediate events that are detected which leads to fewer events sent over the network towards the node processing the top level partial subscription. Additionally, from the logs for the emulation run with the centralized scheme for distributed CEP, DS3 is processing both its sensor data samples for event *C*, but also event *D* from DS4 for the internal partial subscription from Subscription 5.6. The same situation applies for DS5 and DS6: DS6 is processing both its leaf partial subscription for detecting event *E* and the internal partial subscription which matches events *E* and *F*. Moreover, the high selectivity of the internal partial events being processed at nodes DS3 and DS6 means that fewer events are sent from these nodes compared to input events. Consequently, Subscription 5.6 has a very low message overhead. The low probability of detecting complex events for Subscription 5.6 is caused by the internal partial subscriptions high constraints. This also explains the low message overhead used for this subscription.

As opposed to the Subscriptions LC-LS1 and Subscriptions LC-HS1 where partial subscriptions were placed at the same nodes for both

Distributed placement mechanism					
	subscription	event	detected events (probability)	delay (ms)	retransmissions
Subscription 5.3	257	839	1	524	547
Subscription 5.4	283	622	0.6	393894	269
Subscription 5.6	383	77	0.8	89608	142

Table 5.7: Results for high complexity subscriptions with the distributed placement mechanism

placement schemes for distributed CEP, Subscription 5.3 and Subscription 5.4 have many partial subscriptions which must be placed at nodes inside the network. As expected, the distributed placement mechanism seems to find better placement for partial subscriptions considering the results from Table 5.7. The distributed mechanism has a lower message overhead, high probability for detecting complex events and a significantly lower delay for complex event detection compared to the centralized scheme for distributed CEP.

Subscription 5.6 has a higher number of detected complex events which can explain the higher message overhead related to event routing.

5.5.3 Results for various network scenario

In this part of the evaluation, we measure the performance of the distributed placement mechanism for different network scenarios in terms of network density and the mobility speed of the network nodes.

Scenarios 1 and 4 correspond to the network scenario used during the evaluation of the placement mechanisms for subscriptions with varying levels of complexity and selectivity.

The colours red, green, slate-blue respectively represent subscriptions 5.2, 5.3, and 5.4. To avoid network partitions we do not consider scenarios where the network has less than 30 nodes.

The increase in speed of mobility in MANET should lead to a more dynamic topology which causes higher message overhead related to routing. This can have an impact on the performance of higher level communication protocols. Results from Figure 5.5 shows an increase in message overhead for all subscriptions when the speed of mobility increases.

For Subscription 5.3, according to the results from Figure 5.6, the increasing high speed of mobility has no impact on the probability to detect a complex event. In fact, due to the location of DS1, the probability to detect complex events for Subscription 5.3 is not affected by higher speeds of mobility. As mentioned earlier, the data source DS1 is only one hop away

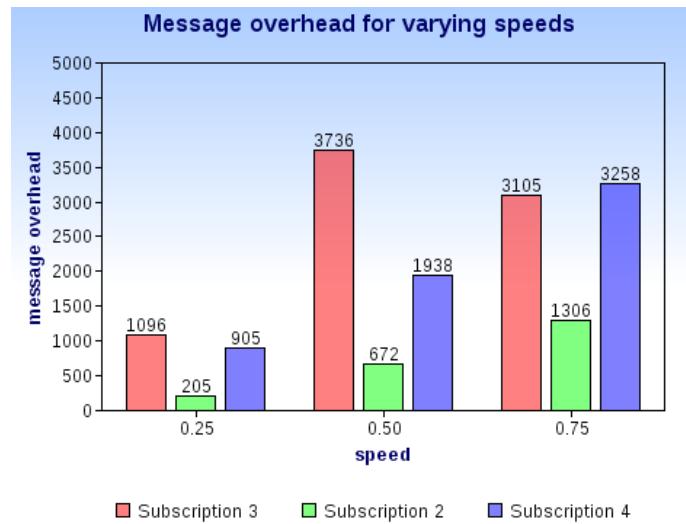


Figure 5.5: Message overhead for varying mobility speeds

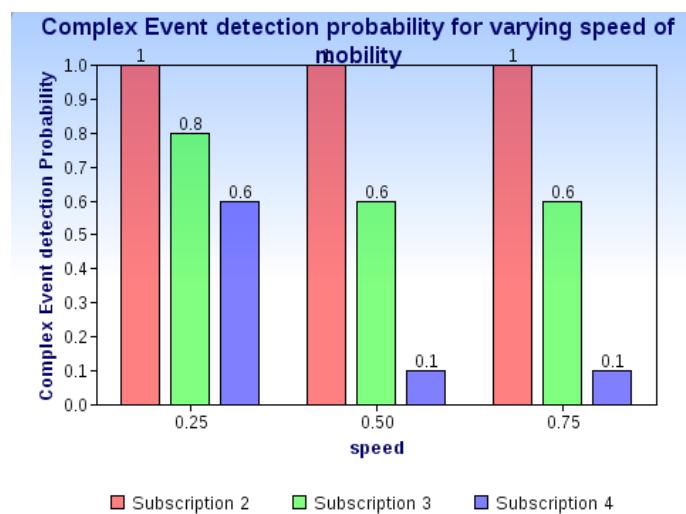


Figure 5.6: Complex event detection probability for varying mobility speeds

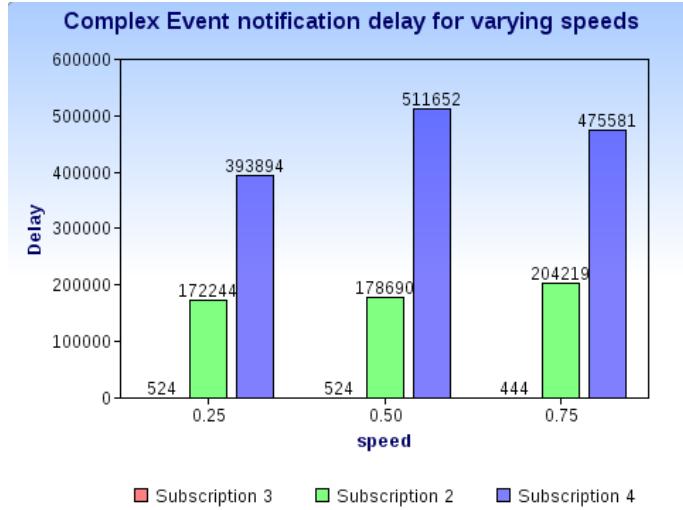


Figure 5.7: Complex event notification delay for varying mobility speeds

from the sink and is static. Additionally, the Subscription 5.3 has very low constraints which allows the CEP at the sink to detect the complex event as soon as it receives event *A* from the DS1.

For Subscription 5.2, the increasing speed of mobility reduce the probability to detect complex events. This is due to the fact that the route between DS3 and the sink is increasingly dynamic which leads to unstable routes and delay. Because the subscription has a high temporal constraints, the increasing delay leads to more complex events being missed. This effect is even higher for Subscription 5.4 which has more complexity in addition to high temporal and spatial constraints.

As expected, results from Figure 5.7 shows that the increasing speed of mobility has no impact the delay of complex event detection for Subscription 5.3. This is caused by the same facts mentioned earlier concerning the probability of detecting the complex event for the same subscription see Figure 5.6.

Additionally, results in Figure 5.7 show a small increase in complex event detection delay for Subscription 5.2. This is due to its low complexity and the fact that only events sent from DS3 are affected by the increase in mobility speed. Consequently, we have a higher delay of complex event detection for Subscription 5.4 which has a significantly higher complexity.

With more nodes in the network, more routes are available for the placement mechanism. This might lead to a higher number of routes between nodes and maybe better execution plans for distributed CEP. For example, log information from emulation run with a network density of 40 nodes shows that the top level partial subscription for Subscription 5.2 is now placed at the data source DS1. This means that there is now a shorter route to data source DS3 through data source DS1.

Results from Figure 5.8, show a decrease of the message overhead for Subscription 5.2, which is due to the fact that a better execution plan was found. Indeed, based on the new execution plan where DS1 is processing

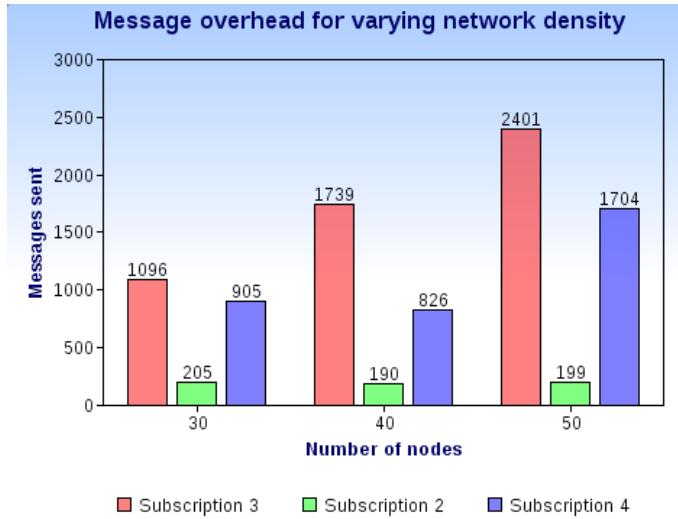


Figure 5.8: Message overhead for varying network density

both the leaf partial subscription for the atomic event A and the internal partial subscription which is matching events A and C , no more events A are sent from DS1. Additionally, the new execution plan suggest that there is a shorter route between DS1 and DS3 which means that events from DS3 are transmitted over a lower number of hops. This also explain the higher probability of detecting complex event for Subscription 5.2 in Figure 5.9.

As Figure 5.10 shows, the delay for detecting the complex event for Subscription 5.2 is also reduced.

5.6 Conclusion

In cases where there are partial subscriptions that are placed inside the network, the distributed placement scheme achieved a lower message overhead and a higher number of complex events detected. For Subscription 5.5 in Table 5.4, the distributed placement scheme has 22% less message overhead than the centralized scheme for distributed CEP and 77% less message overhead compared to the centralized CEP scheme. For Subscription 5.3 in Table 5.7, the distributed placement scheme has 48% less message overhead than the centralized scheme for distributed CEP and 59% less message overhead compared to the centralized CEP scheme. For Subscription 5.4 in the same Table, the distributed placement scheme has 48% less message overhead than the centralized scheme for distributed CEP and 99% less message overhead compared to the centralized CEP scheme. For Subscription 5.6 in Table 5.7, the distributed placement scheme has 43% less message overhead than the centralized scheme for distributed CEP and 85% less message overhead compared to the centralized CEP scheme.

We have also achieved our goal of maintaining a CEP reliability. The distributed placement scheme has higher probability to detect complex events for the subscriptions tested. The mechanism has also kept relatively low

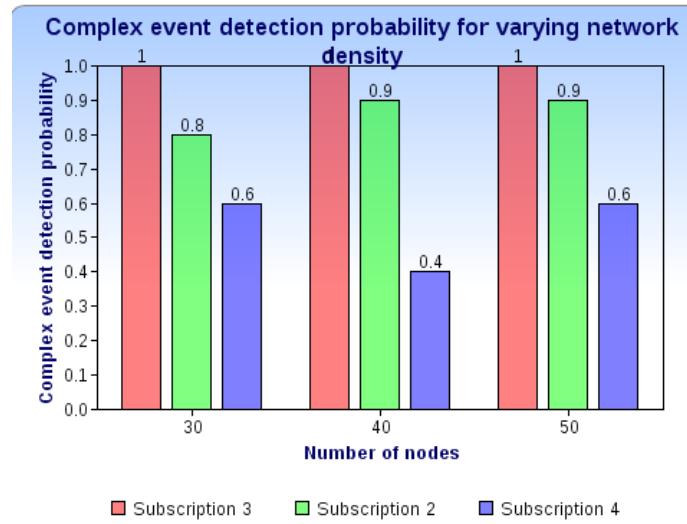


Figure 5.9: Complex event detection probability for varying network density

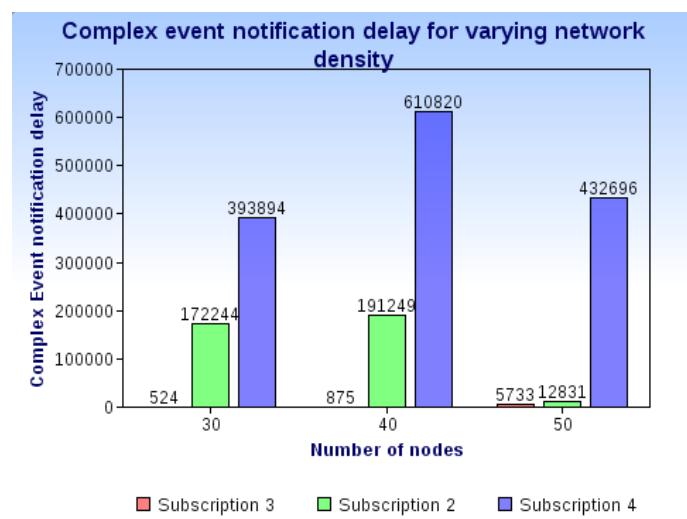


Figure 5.10: Complex event notification delay for varying network density

delay for complex event detection compared to the centralized scheme but did not make any significant improvement (except for some cases like Subscription 5.4 and Subscription 5.2) in that area (this is out of scope for this thesis).

Finding the optimal or near optimal execution plan doesn't guaranty both lower message overhead, low delay and higher complex event detection probability. For example, while the distributed scheme manages to find a better placement (in terms of lower message overhead) for Subscription 5.5 compared to the centralized scheme, the latter has shorter delay for detecting the complex event. This suggest that when determining the cost of an execution plan, the number of hops is not enough if other performance metrics like delay must be optimized as well. This however might not be trivial since techniques used to achieve minimal delay might lead to higher message overhead. For example, techniques like replication are usually used to increase system reliability however, in our case, this might imply redundant processing and data transmission which violates the main purpose of energy conservation through minimal message overhead. The case for Subscription 5.5 happened by chance for the centralized scheme, but shows that considering fewer number of hops for event routing is not enough if other metrics like delay must be taken into consideration.

Figure 5.11 sumries the general trends of performance for Subscriptions 2, 3 and 4. The message overhead appears to increase when the speed of mobility increases. The sudden reduction of message overhead for Subscription 3 is difficult to explain due to the random nature of the current mobility model. However, the general trend for the message overhead when speed of mobility increases is upward. Figure 5.11 shows a weak increase in Complex event notification delay in general, when the speed of mobility increases. Moreover, the probability to detect a complex event decreases when the speed of mobility increases.

When the network density increases, the message overhead also increases. However, for Subscriptions with low complexity, this trend can be in the opposite direction if better placement is found as a result of more nodes to consider for placement and more alternative routes between them. One can expect the message overhead to keep increasing as the network density increases. However, as it appears for Subscription 2 this trend will be closely related to the subscription's complexity, selectivity and the location of its data sources. The delay for Complex event notification increases before decreasing for higher density. One can expect a similar trend when the network density increases and new alternative placement plans are made available. The probability to detect a complex event generally increases when the network density increases due to the availability of new and possibly better placement alternatives. However, the trend related to Subscription 4 shows that this is not always the case.

In general, we can say that we have made a step towards a deeper understanding of the performance of placement strategies for varying

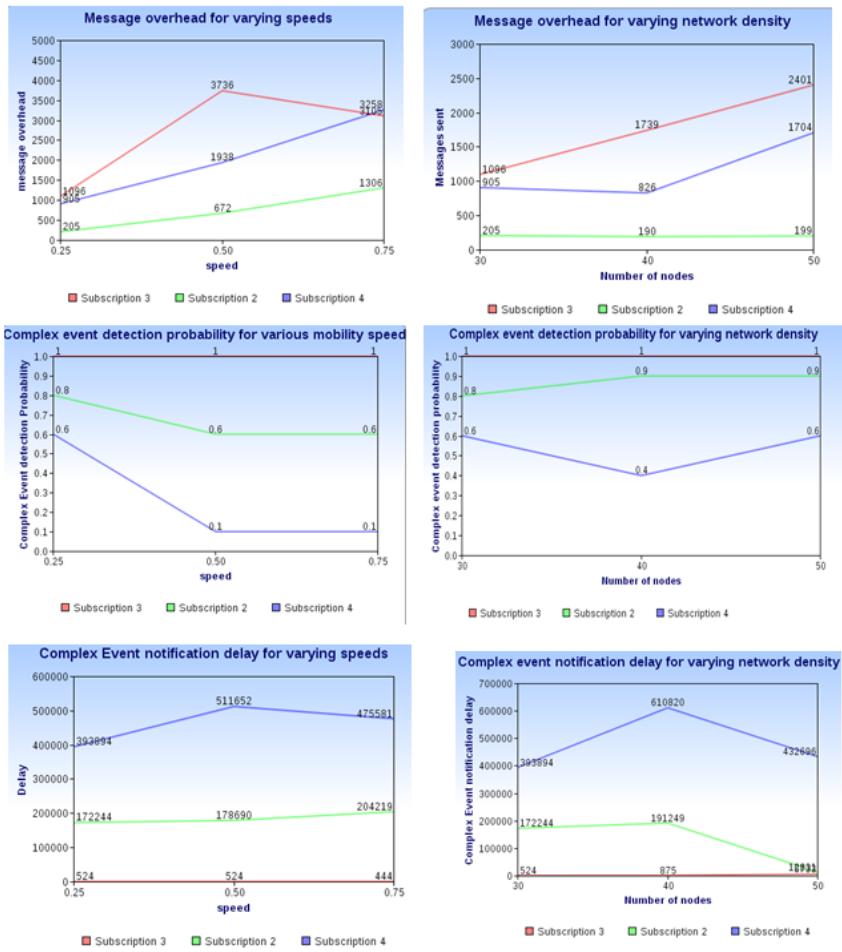


Figure 5.11: Major trends for the performance of the distributed placement mechanism for various network scenarios.

subscription complexity and selectivity on the one hand and varying network scenarios like speed of mobility and density on the other. However, further iteration of evaluation with different system parameter values are necessary in order to confirm the explanations made about the results. The network density parameter values used for the evaluation should be increased in order to have a better view of the trends for the performance of the distributed placement mechanism. The same should be done for the speed of mobility parameter values. More subscriptions should also be used for the evaluation of the distributed placement algorithm for various network scenarios.

Chapter 6

Conclusion

In this part we discuss existing related work and present the contribution of this thesis. A critical analysis of the results and the thesis in general is made. Finally, We suggest further directions beyond the work done in this thesis.

6.1 Related work and contribution of this thesis

To the best of our knowledge no work has been done to deal with the problem of placement for DCEP in MANETs. Existing similar work to the one done in this thesis either focus on the problem of operator placement for in-network processing in static networks or DCEP middleware approaches in sensor networks. While recent work have developed distributed placement approaches, none of them addresses the cases where the network topology is dynamic.

Therefore, the related work for this thesis is grouped into two main classes. The first class comprise work related to enabling DCEP in wireless sensor networks. Some of the related work from this group are: [30, 13] etc... The second class comprises work related to operator placement for in-network processing in wireless sensor networks in general. Some of the related work from this group are: [34, 31, 12, 7] etc...

The first class is related to this thesis in that the goal is to enable DCEP for sensor data processing. The second class is related to this thesis in that both address the task assignment problem for distributed query processing.

In what follows, we present one work from each class since the characteristics related to this thesis are more or less the same in different work done in each group.

In [13] different deployment strategies for a CEP middleware (T-Rex CEP middleware) are developed. Their work is divided into two main parts: the first part is related to the construction of an overlay network consisting of subscription processors (network nodes), and the second part address the problem of how the processors interact during subscription processing and

event routing.

In order to minimize the delay related to complex event notification, the overlay network for subscription processing is a Shortest Path Tree based on the link delay cost metric. This approach uses the TESLA rule language to enable users to express their interest in complex events. However, these rules are partitioned prior to being distributed over the processors that make up the overlay network.

They adopt two different overlay network construction approaches. In the first approach, partial subscriptions are placed on network nodes that form a tree graph with the leaf nodes as data sources. Events are routed from their sources towards to selected root node. The latter is responsible to forward results to the subscribers.

In the second approach, they create multiple trees corresponding to specific subscriptions. In other words, each node that submits a subscription becomes the root of the overlay tree of processors and event routing for that particular subscription. Consequently, events flow from their sources towards the subscriber.

The evaluation for this work compares obtained results to the results from centralized approaches in terms of message overhead and event notification delay.

In [34], a distributed algorithm for operator tree placement is developed. The algorithm assumes no knowledge of network topology and relies on information exchanged between neighbours.

The algorithm has three main stages:

- During the initialization stage, the operator tree for a subscription is flooded inside the network. Each node in the network creates a local state for each operator in the tree representing the cost of producing it.
- In the second stage, neighbouring nodes exchange information about their local states.
- In the third stage, each node updates its local states based on information obtained from the neighbours. If new updates are available, it sends them to the neighbouring nodes.

At any time a node is either forwarding or producing an event based on exchanged state information with its neighbours. The algorithm terminates when no more information is available for exchange.

The algorithm should be able to adapt to topology and cost change while resilient to node or link failure.

The work done in [13] constructs the overlay network for DCEP by flooding the entire network. The goal is to find the optimal placement for partial subscriptions in order to minimize data transmission during event routing

and thus minimize network resource consumption. This is however, both impossible and unnecessary in MANET, due to the dynamic topology. The Shortest Path Tree assumes that network nodes are static which makes it unsuited for MANET. Additionally, it is a waste of resources to use such a high message overhead to find an optimal placement which will most likely be obsolete before event routing begins.

Work done in [34] successfully finds the optimal placement for tree operators at the cost of a high message overhead due to network flooding during initialization stage. This is also unnecessary for the reasons mentioned earlier. Additionally, due to the dynamic nature of the topology in MANETs, there is a risk for a high message overhead related to state information exchange between neighbours. This is due to the fact cost information for a particular operator is related to the location of the node (information obtained from its neighbours), and if it moves away both the old neighbours, itself and its new neighbours might have to exchange new updated information about processing cost for tree operators. Obviously, in a network where all nodes are mobile, this can lead to a flood of update message exchange. Additionally, the algorithm might even never converge since there would be always new state information update.

It appears that these approaches along with those similar to them are not appropriate for DCEP over MANETs. Additionally, to our knowledge, no work related to this thesis has explored placement strategies for DCEP over MANET.

6.2 Critical analysis of the results

In this thesis, we have developed a distributed placement mechanism for DCEP in MANETs. We have also proposed an approach for placement adaptation and replication in order to deal with the execution plan performance deterioration due to the dynamic topology. This provided us with a deeper insight into issues related to distributed placement over a dynamic topology.

We designed an evaluation for the distributed placement mechanism in order to achieve two main goals:

1. First, we wanted to measure the performance of the distributed mechanism compared to the centralized mechanisms developed in earlier work by [17].
2. Second, we wanted to investigate and gain further insight into how the mechanism performs in different network scenarios.

Performance metrics were identified in the light of CEP reliability requirements and identified characteristics, issues and requirements for MANETs, CEP and data processing in sensor networks in general.

In order to evaluate the performance of our distributed placement mecha-

nism for DCEP in different network scenario, we run emulations with different parameter values for network density and the speed of mobility.

Compared to the centralized approaches implemented in [17], the distributed placement mechanism developed in this thesis had a significant reduction in both message overhead and delay for complex event notification. The probability for detecting a complex event was also higher in the distributed placement mechanism approach.

The results related to the performance of the distributed placement mechanism when the speed of mobility increases showed that the message overhead is most negatively impacted compared to the other metrics. During experimentations, higher speed led to issues with finding routes to the data sources, especially the ones that were located further away from the sink.

The results related to the performance of the distributed placement mechanism when the network density increases showed an increase in the probability to detect complex events. Additionally, the message overhead did also decrease in some cases. Results also showed lower event notification delay when the network density increases in some cases.

The performance measurements made for the different network scenarios did not reveal as much as we were hoping to learn. They were highly dependent on specific subscriptions used and in some instances, measurements from different speeds for example were counter intuitive.

6.3 Further work

Network partitioning is quite common in sparse MANETs due to the mobility of the nodes, node failure due to limited resources etc. Therefore, one important feature for a distributed placement mechanism for MANETs is the ability to handle network partitions.

In this thesis we have discussed, designed and implemented an algorithm for placement adaptation. However, in order to evaluate its performance, we would have needed to develop a different network scenario with more nodes. However, we did not have time or the resource to make it. Due to the dynamic nature of the topology in MANETs, the performance of the initial execution plan produced by the placement mechanism is likely to deteriorate. Consequently, placement adaptation is a very important part of a placement mechanism scheme suitable for DCEP over MANETs. In this thesis we have developed a simple algorithm for placement adaptation but more work needs to be done in order to design and implement an efficient adaptation scheme that maintains or even improves the initial execution plan.

The placement mechanism developed in this thesis was rather generic

in that it did not address any specific mobility scenario. Results from a Random Mobility model only show that the mechanism works as it should but more needs to be done in order to make the mechanism useful in a real life scenario. Consequently, one possible direction would be to develop a mobility models classification based on typical mobility patterns from the real world and develop placement mechanisms which are an extension of the one developed in this work and are each specially tuned for a specific mobility model.

Results from the evaluation showed that there is a very close relation between the partial subscription operators characteristics and the performance of the placement mechanism in terms of delay, message overhead and the probability to detect a complex event. Consequently, the splitting component from the middleware developed in [17], should be extended to work closely with the placement component in order to enable more placement optimization based on the subscription operators. We believe there can be high incentives in performing an operator aware distributed placement mechanism.

Context awareness is very important when it comes to data processing over MANET. With context we mean information about the current network topology, link state, mobility speed, number of neighbours, etc. All these factors can have a significant impact on the performance of the placement mechanism and other components of the middleware as well (the communication component for example). Consequently, a placement mechanism should take them into consideration if it is to achieve satisfactory results. However, due to the amount and complexity of context information, it would be inappropriate to make its management a part of the placement component. Instead, a context awareness component could be added to the middleware in order to access, manage and provide context information to the placement component (and possibly other components as well).

Bibliography

- [1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002.
- [2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002.
- [3] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537 – 568, 2009.
- [4] Panayiotis Andreou, Demetrios Zeinalipour-Yazti, Andreas Pamboris, Panos K. Chrysanthis, and George Samaras. Optimized query routing trees for wireless sensor networks. *Inf. Syst.*, 36(2):267–291, April 2011.
- [5] Fan Bai and Ahmed Helmy. A survey of mobility models in wireless adhoc networks. In *Wireless Ad Hoc and Sensor Networks*. Springer, 2006.
- [6] S.H. Bokhari. A shortest tree algorithm for optimal assignments across space and time in a distributed processor system. *Software Engineering, IEEE Transactions on*, SE-7(6):583–589, 1981.
- [7] Boris Jan Bonfils and Philippe Bonnet. Adaptive and decentralized operator placement for in-network query processing. In *Proceedings of the 2nd international conference on Information processing in sensor networks*, IPSN’03, pages 47–62, Berlin, Heidelberg, 2003. Springer-Verlag.
- [8] Luciano Bononi, Marco Conti, and Lorenzo Donatiello. A distributed mechanism for power saving in ieee 802.11 wireless lans. *MONET*, 6(3):211–222, 2001.
- [9] Alejandro P. Buchmann and Boris Koldehofe. Complex event processing. *it - Information Technology*, 51(5):241–242, 2009.
- [10] Jan Carlson. *Event Pattern Detection for Embedded Systems*. PhD thesis, Mälardalen University, Department of Computer Science and Electronics, 2007.

- [11] Hakima Chaouchi. *The Internet of things:connecting objects to the web*. ISTE/Wiley, 2010.
- [12] Georgios Chatzimilioudis, Nikos Mamoulis, and Dimitrios Gunopoulos. A distributed technique for dynamic operator placement in wireless sensor networks. In *Proceedings of the 2010 Eleventh International Conference on Mobile Data Management*, MDM '10, pages 167–176, Washington, DC, USA, 2010. IEEE Computer Society.
- [13] Gianpaolo Cugola and Alessandro Margara. Deployment strategies for distributed complex event processing. *Computing*, 95(2):129–156, 2013.
- [14] Thaddeus O Eze and Mona Ghasseman. Heterogeneous mobility models scenario: Performance analysis of disaster area for mobile ad hoc networks.
- [15] Ricki G. Ingalls. Introduction to simulation: introduction to simulation. In *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*, WSC '02, pages 7–16. Winter Simulation Conference, 2002.
- [16] Søberg Jarle. *CommonSens : A Multimodal Complex Event Processing System for Automated Home Care*. PhD thesis, Faculty of Mathematics and Natural Sciences, University of Oslo, 2011.
- [17] P. Kamisinski, V. Goebel, and T. Plagemann. A reconfigurable distributed cep middleware for diverse mobility scenarios. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 615–620, 2013.
- [18] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2005.
- [19] Natallia Kokash. An introduction to heuristic algorithms. *Department of Informatics and Telecommunications*, 2005.
- [20] Christoph Lameter. Cgroups. <https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt>. Accessed: 2013-09-15.
- [21] Averill Law. *Simulation Modeling and Analysis (McGraw-Hill Series in Industrial Engineering and Management)*. McGraw-Hill Science/Engineering/Math, 2006.
- [22] Mohammad Llyas, editor. *The handbook of ad hoc wireless networks*. CRC Press, 1st edition, 2002.
- [23] Zongqing Lu and Yonggang Wen. Distributed and asynchronous solution to operator placement in large wireless sensor networks. In *Mobile Ad-hoc and Sensor Networks (MSN), 2012 Eighth International Conference on*, pages 100–107, 2012.

- [24] Zongqing Lu, Yonggang Wen, Rui Fan, Su-Lim Tan, and J. Biswas. Toward efficient distributed algorithms for in-network binary operator tree placement in wireless sensor networks. *Selected Areas in Communications, IEEE Journal on*, 31(4):743–755, 2013.
- [25] David C. Luckham and Brian Frasca. Complex event processing in distributed systems. Technical report, Stanford University, 1998.
- [26] Alessandro Margara and Gianpaolo Cugola. Processing flows of information: from data stream to complex event processing. In *Proceedings of the 5th ACM international conference on Distributed event-based system*, DEBS ’11, pages 359–360, New York, NY, USA, 2011. ACM.
- [27] Anu Maria. Introduction to modeling and simulation. In *Proceedings of the 29th conference on Winter simulation*, WSC ’97, pages 7–13, Washington, DC, USA, 1997. IEEE Computer Society.
- [28] Dr. Michael Pidwirny and Scott Jones. Fundamentals of physical geography (2nd edition). <http://www.physicalgeography.net/fundamentals/4b.html>. Accessed: 2013-09-20.
- [29] Udo W. Pooch and James A. Wall. *Discrete event simulation: a practical approach*. CRC Press, Inc., Boca Raton, FL, USA, 1993.
- [30] O. Saleh and K.-U. Sattler. Distributed complex event processing in sensor networks. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 2, pages 23–26, 2013.
- [31] Utkarsh Srivastava, Kamesh Munagala, and Jennifer Widom. Operator placement for in-network stream query processing. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS ’05, pages 250–258, New York, NY, USA, 2005. ACM.
- [32] C. Puttamadappa Subir Kumar Sarkar, T.G. Basavaraju. *Ad Hoc Mobile Wireless Networks: Principles,Protocols, and Applications*. Auerbach Publication, 1st edition, 2012.
- [33] U. Westermann and R. Jain. Toward a common event model for multimedia applications. *MultiMedia, IEEE*, 14(1):19 –29, jan.-march 2007.
- [34] Lei Ying, Zhen Liu, D. Towsley, and C.H. Xia. Distributed operator placement and data caching in large-scale sensor networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 977–985, 2008.
- [35] Jun-Hu Zhang and Feng-Jing Shao. Bf-k: a near-optimal operator placement algorithm for in-network query processing. *JNW*, 5(10):1118–1126, 2010.